Predicting Heterogeneity and Serverless Principles of Converged High-Performance Computing, Artificial Intelligence, and Workflows

Pedro Bruel, Sai Rahul Chalamalasetti, Aditya Dhakal, Eitan Frachtenberg, Ninad Hogade, Rolando Pablo Hong Enriquez, Alok Mishra, Dejan Milojicic, Pavana Prakash, and Gourav Rattihalli, Hewlett Packard Labs

Digital Object Identifier 10.1109/MC.2023.3332973
Date of current version: 5 January 2024

Traditional high-performance computing and modern artificial intelligence computing are converging with workflows as a common paradigm. We predict nine principles of heterogeneity and serverless computing for this convergence, from high-level programming to low-level hardware.

High-performance computing (HPC) is increasingly converging with artificial intelligence (AI), and both are increasingly

0018-9162/24©2024IEEE

EDITOR **DEJAN MILOJICIC**
Hewlett Packard Labs;
dejan.milojicic@hpe.com

expressed as workflows. Workflows enable a higher level of abstraction that is easier to develop, (re)use, and operate. Both HPC and AI depend heavily on accelerators, and they both adopt serverless computing. Similarly to workflows, serverless computing also raises the level of abstraction and simplifies DevOps.[1] In addition, it matches the fine granularity of accelerators in terms of time and size; they intuitively represent a good match in terms of performance and utilization.

When analyzing this convergence, three perspectives with different requirements and benefits exist:

1. End users care about latency or throughput of workflows at scale and ease/convenience of use.
2. Developers care about ease of development, for example, constructing workflows from existing workloads and making quality-of-service (QoS) guarantees.
3. Providers (of services infrastructure) primarily care about meeting service-level agreements for user QoS and maximizing infrastructure utilization.

These three roles intertwine, and individuals could easily play two or even all three roles. For example, an end user of some services can be a provider to other users, and a developer can conduct operations.

The principles and approaches we describe strive toward enabling seamless scalability and fluidity for end users, increased productivity of developers, and improved performance efficiency of providers.

To navigate these principles, we organized them in Figure 1, listed in order of description. Each principle

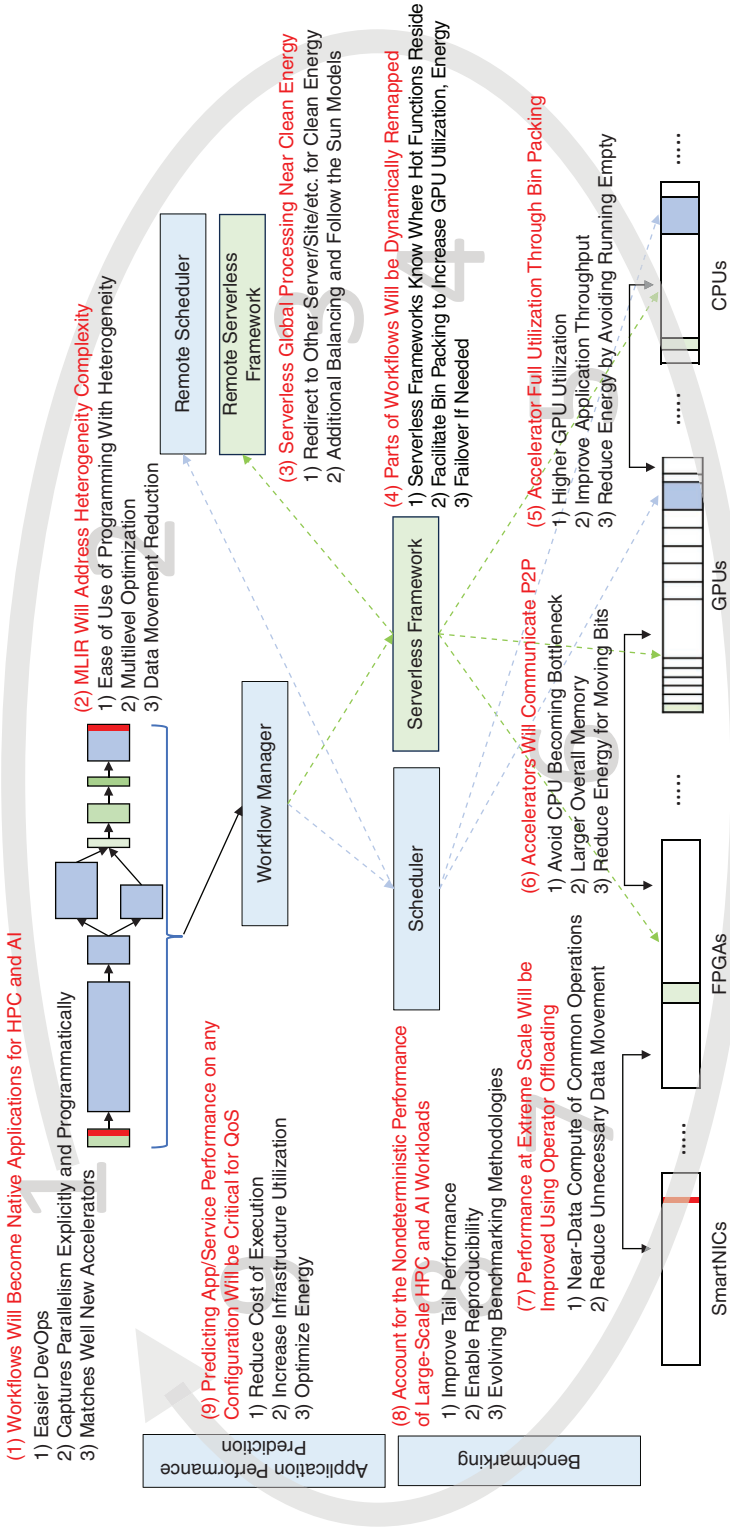

FIGURE 1. Landscape of principles. MLIR: Multilevel Intermediate Representation; P2P: peer to peer; FPGA: field-programmable gate array; SmartNIC: smart network interface card.

enumerates possible benefits. The figure should be read clockwise, starting with principle 1.

WORKFLOWS WILL BECOME NATIVE APPLICATIONS FOR HPC AND AI

With no intention of exploring the history of HPC and while focusing on the software evolution over hardware innovations, we would like to recall simpler times when calculations were performed on isolated, domain-specific problems by homogeneous CPU-based hardware. As the computational problems grew in complexity, they were typically broken into smaller tasks or files. However, it was still feasible then for a single programmer or small team to completely rewrite the code. Today's variety of domain-specific workflows in HPC has pushed application scientists and engineers to propose various workflow-management systems with no clear standards or implementation patterns beyond a few workflow specifications and programming languages (Figure 2).

As individual applications become workflow friendly by embedding CPU/GPU programming in single executables, it's becoming clearer that these partial solutions are unlikely to stay relevant, partially because the end of the Dennard-scaling era is leading to accelerator diversification. Creating truly native solutions for HPC will likely involve not only methodologies from multiple domains of knowledge at once, but also 1) distributing computing tasks efficiently among emerging accelerators, 2) dynamically redeploying at scale, and 3) generating/testing workflows interactively using low-code programming models and simulations. The need for human experts to achieve next-level performance on these increasingly complex computational workflows will be complemented by machine learning and AI. These techniques will not only be used within the workflows themselves but also as part of the HPC infrastructure. Future HPC is evolving toward harmoniously engineered workflows whose complexity can be abstracted while still performing optimally under flexible conditions.

Optimally deploying these heterogeneous codes will require attention to how these codes are represented, as discussed next.

MULTILEVEL INTERMEDIATE REPRESENTATION WILL ADDRESS HETEROGENEITY COMPLEXITY

As applications become more workflow centric, it becomes increasingly challenging for programming languages and compilers to ensure efficient task execution, unified representation, interoperability, and good abstraction. Multilevel Intermediate Representation (MLIR)[2] is an open source project initiated by LLVM to develop a new intermediate representation for compilers. It addresses some of the limitations of compilers that use traditional IRs (like LLVM-IR) by providing a more expressive, flexible, and efficient way to represent program structures in the front end. It also enables efficient compilation, optimization, and interoperability across diverse programming languages, domains, and hardware platforms, fostering innovation and collaboration in compiler design, AI, HPC, and more.

MLIR offers several advantages to HPC, AI, and workflow optimization.

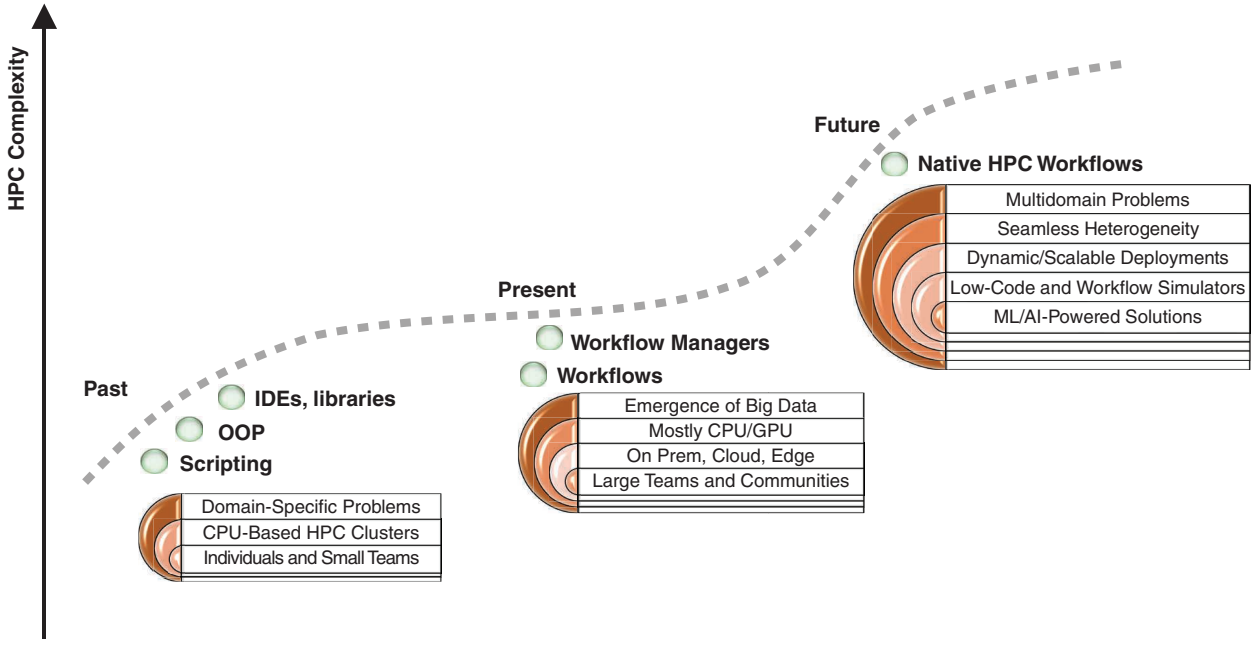

FIGURE 2. Evolution of workflows. IDE: integrated development environment; OOP: object-oriented programming; ML: machine learning workflow.

Its robust optimization capabilities, including high-level optimizations like data-layout transformation, alongside low-level optimizations, such as loop unrolling and vectorization, allow performance tuning of HPC applications. Its hardware-agnostic representation allows workflows to adapt effortlessly to heterogeneous architectures, like CPUs, GPUs, field-programmable gate arrays (FPGAs), or future quantum computing accelerators, enabling customized optimization for various HPC configurations. It provides a unified representation for complex workflows, including HPC and AI components, ensuring easy integration and optimization across multiple computing workloads in a single format. Its framework-specific dialects, which support TensorFlow, PyTorch, etc., ensure seamless model translation and integration across various components of AI processes, providing compatibility and ease of implementation.

Moreover, the flexibility and dynamic compilation capabilities of MLIR aid in sustainable workflow scheduling (discussed next) and enable dynamic deployment of HPC/AI workflows (principles 3 and 4), enabling real-time optimizations and efficient scaling of individual workloads.

SERVERLESS GLOBAL PROCESSING NEAR CLEAN ENERGY

In the drive toward a more sustainable digital realm, the potential of workflow scheduling emerges as a transformative force. Through the decomposition of workflows into smaller, manageable functions, an opportunity arises to strategically deploy these functions based on geographical and environmental considerations. Geodistributed computational resources differ in their energy sourcing. Some benefit from renewable energy, while others draw power from carbon-intensive fossil fuels; some have faster computing capabilities, and vice versa. By leveraging workflow scheduling, functions can be located based on utilization, operating cost, and, critically, environmental sustainability.[3,4] This inherent flexibility allows functions within workflows to run in varied geographical regions, aiming to either minimize carbon emissions or optimize the total runtime of a workflow.

To validate this idea, we proposed a framework (shown in Figure 3) and conducted a series of experiments. We used Globus Compute[5] as the backbone of our framework to distribute functions across various geodistributed computing resources. Our experimental setup encompassed a variety of computational resources: a Kubernetes cluster at Hewlett Packard Enterprise (HPE) in Milpitas, CA, USA, and two Google Cloud Platform servers, one in Los Angeles, CA, and another in Council Bluffs, IA. Each location had its distinct energy profile, influencing the function deployment strategies. The functions originated in HPE's office in Fort Collins, CO. The global workflow scheduler was interfaced with Performance Co-Pilot (PCP) to monitor crucial metrics, such as power consumption. Our experimental focus was the implementation of a carbon-aware scheduling policy aimed at minimizing carbon emissions in function deployment. Utilizing the proposed policy, the system was able to execute more functions (utilize more hardware) with lower carbon emissions.

Building upon the operational advantages of geodistributed scheduling, we next explore dynamic redeployment, harnessing heterogeneous hardware to further optimize our global workflow framework.

PARTS OF WORKFLOWS WILL BE DYNAMICALLY DEPLOYED

With the increase in HPC and AI in computational research, the need to alleviate the bottleneck of statically deployed workflows grows urgent. Traditionally, workflows have been hosted on a fixed number of machines, resulting in resource underutilization. The growth of cloud-based virtual machines and bare-metal nodes enabled a game-changing solution: dynamic (re) deployment of workflow components. This strategy ensures that specified components of a workflow operate on specialized hardware, maximizing utilization and decreasing workflow execution runtime.[6] These components can be dynamically redeployed on specific hardware accelerators when they are available.

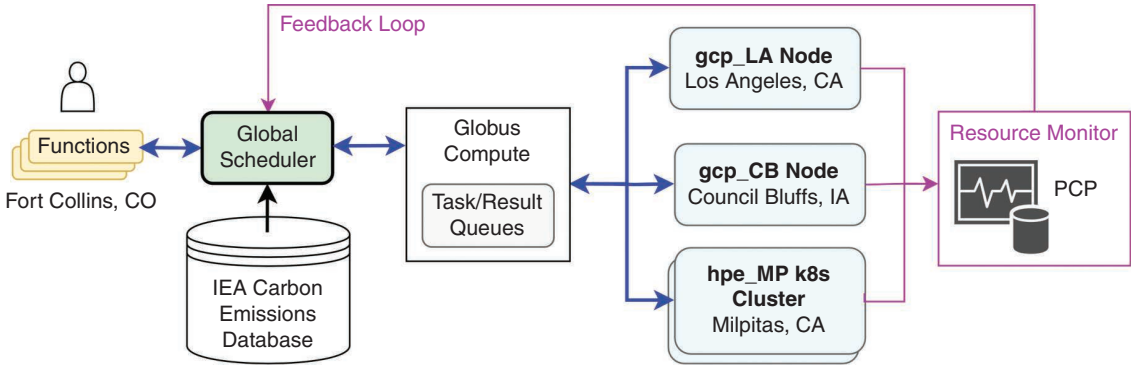

FIGURE 3. Carbon-aware scheduling of functions within a workflow. IEA: International Energy Agency; PCP: Performance Co-Pilot.

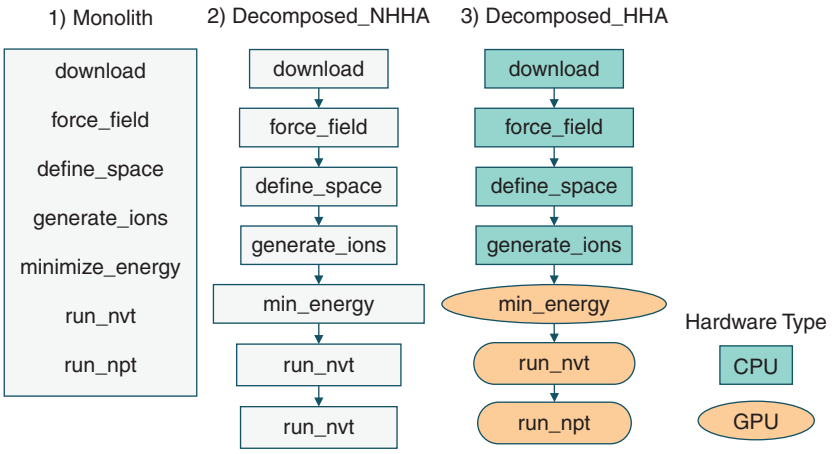

FIGURE 4. Three distinct execution techniques for the GROMACS Lysozyme workflow.

To validate this dynamic redeployment model, we used the GROMACS Lysozyme workflow.[7] We tried three different execution methods:

1. *Serial monolithic execution (monolith)*: This is the traditional approach, executing the workflow as serial, monolithic tasks.
2. *Decomposed but heterogeneity-oblivious execution (decomposed_NHHA)*: The workflow was decomposed into its constituent functions and executed on a serverless platform without specific hardware considerations.
3. *Decomposed and heterogeneous hardware-aware execution (decomposed_HHA)*: Enhancing the second method, this technique dynamically redeployed the decomposed functions, assigning intensive tasks to specialized nodes with GPUs (when available).

Figure 4 details these execution techniques. Monolith and decomposed_NHHA are inefficient since they don't utilize the specialized hardware, thus taking longer to run. While decomposed_NHHA has extended execution times caused by container cold starts, the decomposed_HHA method significantly reduces runtime.

Dynamically redeploying workflow components optimizes task execution on suitable hardware, ensuring enhanced cluster utilization and minimized runtime. This shift toward dynamic deployment signifies a future of optimized resource allocation in computational research.

There is a huge demand for GPUs nowadays, shifting importance from user workload execution to maximizing GPU utilization, which leads us to the next principle.

ACCELERATOR FULL UTILIZATION THROUGH BIN PACKING

A workflow task might not saturate the entire GPU, so exploiting accelerator granularity could be increasingly important for HPC.[8,9] To motivate finer accelerator granularity, we present an experiment where we ran the same nano-LAMMPS workflow with different GPU partition sizes. We picked a kernel that ran more than 6,000 times during the workflow execution and plotted its runtime with varying amounts of GPU compute (GPU percentage) in Figure 5(a). We can see the runtime of some runs (0–1,000) improve when the GPU percentage gets higher, although not as much between kernels running at 50% GPU and 100% GPU. However, the runtime of almost all kernels hovers around 5 µs and does not change regardless of the GPU percentage. These

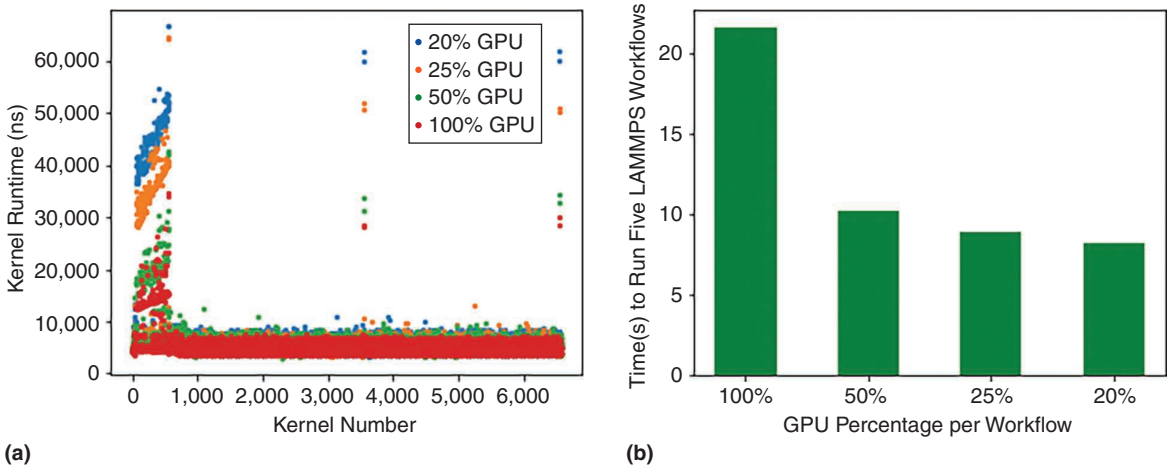

FIGURE 5. (a) Kernel runtime of workflow across different GPU percentages. (b) Completion time of five workflows running concurrently.

measurements show that the kernels in these workflows do not require 100% GPU, and often 20% GPU suffices. Packing the GPU with multiple workflows, each getting a certain GPU percentage, could increase the GPU throughput.

In another experiment, we look at the throughput of a bin-packed GPU compared to a situation where the GPU is not multiplexed. In Figure 5(b), we present the time to run five LAMMPS workflows. Giving each workflow its own "bin" with 20% GPU completes running all workflows 60% faster than running each workflow individually with 100% GPU. This reduction in makespan stems from the increased throughput due to partitioning the GPU and running workflows concurrently.

Data transfer across computing elements [CPUs, GPUs, FPGAs, smart network interface cards (SmartNICs), etc.] is the slowest part of such workflows. Therefore, optimizing this communication is imperative. The next principle discusses optimizing performance with peer-to-peer (P2P) communications.

ACCELERATORS WILL COMMUNICATE P2P

When accelerators consume the majority of an application's computation, it is necessary to enable faster data movement to them; this currently uses the Peripheral Component Interconnect Express (PCIe) interface. However, the number of PCIe lanes at the CPU socket level limits the number of accelerators at a node level. Although a dual-socket-based server allows more accelerators per server, the NUMA-node connectivity interface for communication between accelerators across sockets can become a bottleneck. This is where P2P access to accelerators can help.

To illustrate this technique, we measured transfers from a Mellanox InfiniBand (IB) 200 Gb/s network interface controller (NIC) to a Nvidia A100 GPU using GPUDirect,[10] which uses PCIe P2P transfers. We compared it to a host bounce back (the host buffer as an intermediate data copy). We used OpenMPI 4.1.1, which supports GPUDirect transfer and point-to-point OSU latency and bandwidth (BW) benchmarks for the analysis.[11]

Figure 6(a) shows that GPU-to-GPU device communication across nodes through the IB NIC using GPUDirect/P2P exhibits a 1.2–3 times higher BW than host bounce back. Also, the GPUDirect/P2P GPU device buffer transfer could saturate the PCIe interface to the practical BW limit of 24 GB/s (75% of the theoretical PCIe Gen4 ×16 BW of 32 GB/s). In addition, GPUDirect/P2P [Figure 6(b)] also exhibits a 1.2–5 times lower latency for message sizes below 8 MB, but for message sizes larger than 16 MB the latency of GPUDirect/P2P transfers converges to the host bounce-back transfer latency.

With machine learning training workloads that rely on GPU-to-GPU communication, a higher P2P transfer BW allows faster training speeds. For HPC workloads, most of the GPU-to-GPU communication would use small message sizes, so lower latency using P2P transfer will reduce the overall application execution time.

In addition to GPU partitioning and P2P optimizations, the next principle introduces operator offloading as another important performance optimization.

PERFORMANCE AT EXTREME SCALE WILL BE IMPORVED USING OPERATOR OFFLOADING

In addition to hardware accelerators, distributed HPC and AI workflows rely on industry-standard libraries, such as the Message Passing Interface (MPI), to effectively distribute, perform, and synchronize computation across interconnected machines. Common distributed operations, such as synchronizing data buffers in multiple machines via an aggregation operation, are encapsulated in MPI *collective communication routines*, or collectives. The operation *AllReduce* collective in MPI parlance is a fundamental operation of AI training workloads and also appears in many HPC workloads.[12]

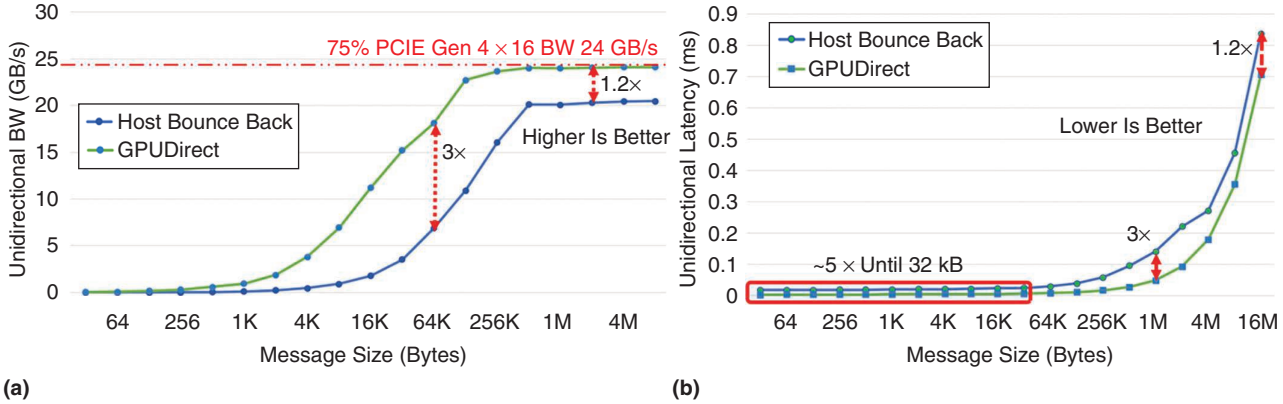

FIGURE 6. (a) Unidirectional internode GPU-to-GPU BW. (b) Unidirectional internode GPU-to-GPU latency.

Driving these critical communication operations using the CPU or an accelerator, such as the GPU, consumes valuable computation and memory bandwidth, slowing down the application performance.[13] Other than MPI collectives, operators, such as data sorting, filtering, and encrypting/decrypting,[14] are also ubiquitous in HPC and AI workflows. Since all of these critical and high-frequency operators depend on network communication, freeing CPU and GPU BW by offloading operators to network hardware, such as NICs and switches, presents a great opportunity to improve the performance of HPC and AI workflows.[15]

When pushing AI and HPC workflows toward extreme scales, for example, systems with tens or hundreds of thousands of GPUs, it becomes extremely hard to hide communication operations behind computation operations. Figure 7 presents the results of analytical simulations and modeling of the distributed training of convolutional neural networks (CNNs) and large language models (LLMs) in extreme-scale systems. The figure highlights the large impact that driving communication has on GPU bandwidth, especially when increasing the message size of MPI collectives. We modeled analytically and simulated the use of AllReduce operations in AI workflows without NIC offloading (shown in green lines), with NIC offloading (shown in blue lines), and with NIC offloading plus full gradient caching (red lines). This last scenario is impractical in real hardware since the necessary

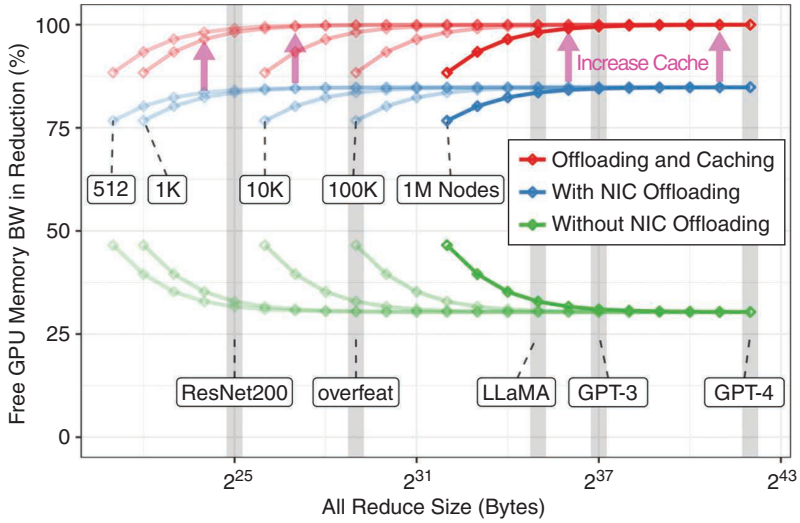

FIGURE 7. Analytical simulations and modeling of distributed training of CNNs and LLMs in extreme-scale systems, highlighting the large impact that driving communication has on GPU bandwidth.

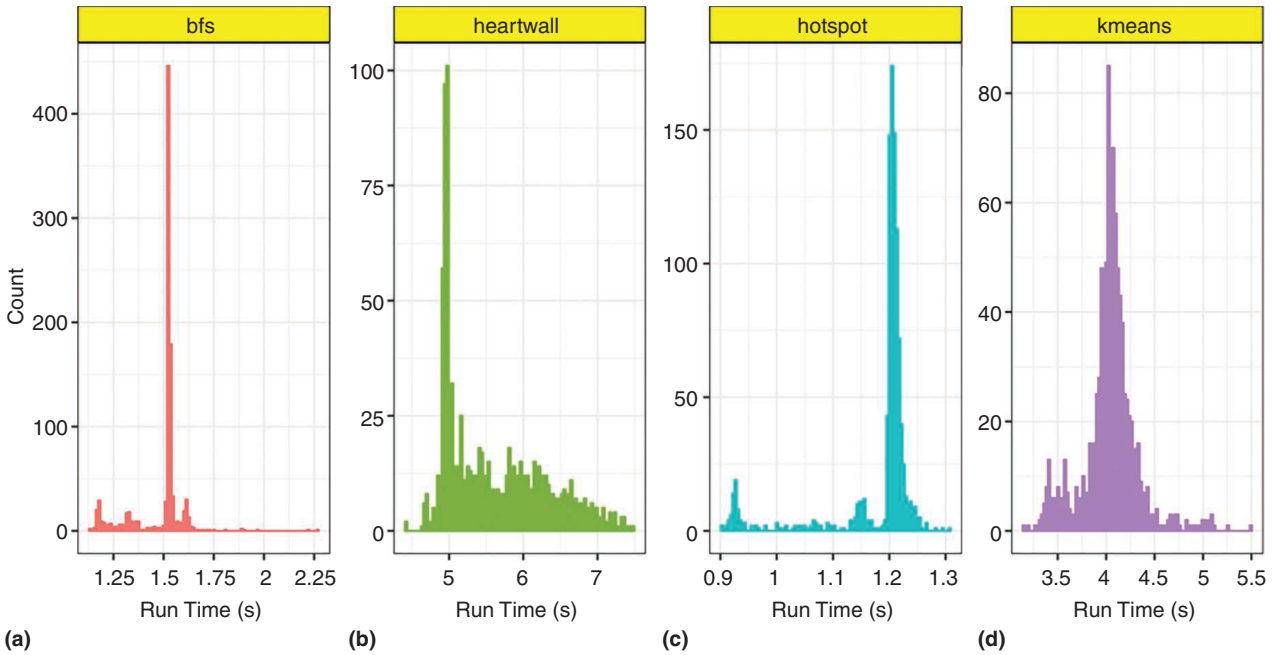

FIGURE 8. Performance histograms for four example applications from the Rodinia HPC benchmark suite, run 1,000 times each on a single machine with no interference. These applications represent distributions that are (a) approximately near constant, (b) right tailed, (c) left-tailed, and (d) symmetric.

cache size would be too expensive, but it can serve as a basis for comparison for increasing NIC cache sizes from the feasible scenario shown on the blue lines. We simulated real AI training workloads, from ResNet200 to GPT-4, highlighting the total AllReduce size involved in training each neural network. These simulations and models demonstrate that, after saturation, driving communication during training would leave only around 30% of GPU memory BW free for computation, while offloading the AllReduce to a capable NIC would leave up to 87% GPU memory BW free, which represents an expressive amount of computing power at extreme scales.

The seven optimization mechanisms and operation principles described so far emphasize heterogeneity in hardware in software, which complicates performance evaluation, as discussed next.

ACCOUNT FOR THE NONDETERMINISTIC PERFORMANCE OF LARGE-SCALE HPC AND AI WORKLOADS

The combination of large-scale and heterogeneous software, middleware, and hardware means that every time we measure system performance we could be getting a different result (as shown for illustration in Figure 8). Some variability could be reduced or controlled, but likely not all of it. If the observed performance differences are relatively large and unpredictable, this nondeterministic behavior obfuscates the actual performance of the underlying system. It therefore becomes increasingly harder to answer critical business questions, such as: Does system A perform better than system B? What is the cost/performance of a system? Did its performance regress or improve?

The key to answering such questions is to handle performance like every other nondeterministic factor using statistical tools for distributions, similar to the research tools used in social and medical sciences. These tools can range from simple hypothesis testing and quantification of uncertainty to more advanced topics, such as divergence metrics and causality analysis. Although these tools carry an implicit penalty, both in additional work and additional expertise, they also carry the promise of better performance reproducibility, correct interpretations, and actionable insights.

PREDICTING APP/SERVICE PERFORMANCE ON ANY CONFIGURATION WILL BE CRITICAL FOR QOS

The nondeterministic behavior of complex system performance brings to light the need for accurate estimates of performance and its distribution. For example, performance models of key applications have always been critical in the design and procurement of future computer systems,[16] but these models often assume a homogeneous workload and architecture.

As another example, when making scheduling decisions on a shared cluster, understanding the expected tail and outliers of the performance distribution of an application can impact the timing of its scheduling to maintain service-level agreements for other jobs.

Together with collaborator Izzat el Hajj at the American University of Beirut (AUB), we have been developing performance prediction mechanisms based on a machine learning model trained on low-level performance metrics. This model has been successfully demonstrated in tasks such as predicting the performance of known applications on new hardware configurations,[17] which can be applied to the problem of selecting new hardware without benchmarking on all available choices. These techniques were also successful in predicting when applications are near the end of their execution as a very useful prediction for supercomputer and mission-critical schedulers.

We presented nine principles of convergence of HPC, AI, and workflows. These end-to-end principles cover workflows through middleware to hardware. Nevertheless, there are many other missing aspects where this convergence can apply.[18,19,20] We did not even touch on nonfunctional aspects, such as security, reliability, scale, availability, etc. Each of these represents a considerable challenge but also an opportunity for improved usability, development, and delivery of converged HPC, AI, and workflows.

ACKNOWLEDGMENT

We would like to thank our many collaborators who graciously contributed to different aspects of this work: Wen-mei Hwu and Deming Chen of the University of Illinois Urbana-Champaign, Gustavo Alonso of ETH, Izzat El Hajj of AUB, Ian Foster of the University of Chicago, Avi Mendelson of Technion, and Sudeep Pasricha of Colorado State University. We would also like to thank Alex Qi, Alex Weaver, Hongzheng Tian, Kaiwen Cao, Kanchu Kiran, Liad Gerstman, Philipp Raith, Vijay Thurimella, and Viyom Mittal for contributing to some of the insights.

REFERENCES

1. L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of DevOps concepts and challenges," *ACM Comput. Surv. (CSUR)*, vol. 52, no. 6, pp. 1–35, Nov 2019, doi: 10.1145/3359981.
2. C. Lattner et al., "MLIR: Scaling compiler infrastructure for domain specific computation," *IEEE/ACM Int. Symp. Code Gener. Optim. (CGO)*, 2021, pp. 2–14, doi: 10.1109/CGO51591.2021.9370308.
3. S. Qi, D. Milojicic, C. Bash, and S. Pasricha, "SHIELD: Sustainable hybrid evolutionary learning framework for carbon, wastewater, and energy-aware data center management," in *Proc. IEEE Int. Green Sustain. Comput. Conf.*, 2023.

4. C. Bash, N. Hogade, D. Milojicic, G. Rattihalli, and C. D. Patel, "Sustainability: Fundamentals-based approach to paying it forward," *Computer*, vol. 56, no. 1, pp. 125–132, Jan. 2023, doi: 10.1109/MC.2022.3219173.
5. "Computing with Globus." Globus. Accessed: Jul. 11, 2023. [Online]. Available: https://www.globus.org/compute
6. G. Rattihalli et al., "Fine-grained heterogeneous execution framework with energy aware scheduling," in *Proc. IEEE 16th Int. Conf. Cloud Comput. (CLOUD)*, 2023, pp. 35–44, doi: 10.1109/CLOUD60044.2023.00014.
7. "Lysozyme in water." MD Tutorials. Accessed: Jul. 11, 2023. [Online]. Available: http://www.mdtutorials.com/gmx/lysozyme/
8. T. Pfandzelter et al., "Kernel-as-a-Service: A serverless programming model for heterogeneous hardware accelerators," in *Proc. ACM Middleware*, 2023, pp. 1–15.
9. A. Dhakal et al., "Fine-grained accelerator partitioning for Machine Learning and Scientific Computing in Function as a Service Platform," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage, Anal.*, New York, NY, USA, Denver, CO, USA: ACM, Nov. 12–17, 2023, pp. 1606–1613, doi: 10.1145/3624062.3624238.
10. "Enhancing data movement and access for GPUs." Nvidia Developer. Accessed: Oct. 11, 2023. [Online]. Available: https://developer.nvidia.com/gpudirect
11. "MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, RoCE, and Slingshot." MVAPICH. Accessed: Oct. 11, 2023. [Online]. Available: https://mvapich.cse.ohio-state.edu/benchmarks/
12. S. Chunduri et al., "Characterization of MPI usage on a production supercomputer," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2018, pp. 386–400, doi: 10.1109/SC.2018.00033.
13. S. Rashidi et al., "Enabling compute-communication overlap in distributed deep learning training platforms," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2021, pp. 540–553, doi: 10.1109/ISCA52012.2021.00049.
14. D. Korolija et al., "Farview: Disaggregated memory with operator off-loading for database engines," in *Proc. 12th Annu. Conf. Innov. Data Syst. Res.*, 2022, pp. 1–14.
15. T. Hoefler. *General in-Network Processing - Time is Ripe!* (Oct. 1, 2020). Accessed: Jul. 11, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=t6jdjnnIRZs
16. D. J. Kerbyson, H. J. Alme, A. Hoisie, F. Petrini, H. J. Wasserman, and M. Gittings, "Predictive performance and scalability modeling of a large-scale application," in *Proc. ACM/IEEE Conf. Supercomput.*, 2001, pp. 1–37, doi: 10.1145/582034.582071.
17. A. Nassereldine et al., "Predicting the performance-cost trade-off of applications across multiple systems," in *Proc. IEEE/ACM 23rd Int. Symp. Cluster, Cloud Internet Comput. (CCGrid)*, 2023, pp. 216–228, doi: 10.1109/CCGrid57682.2023.00029.
18. R. M. Badia, I. Foster, and D. Milojicic, "More real than real: The race to simulate everything," *Computer*, vol. 55, no. 7, pp. 67–72, Jul. 2022, doi: 10.1109/MC.2022.3173359.
19. N. Dube, P. Faraboschi, D. Milojicic, and D. Roweth, "Future of HPC: Internet of workflows," *IEEE Internet Comput.*, vol. 25, no. 5, pp. 26–34, Sep./Oct 2021, doi: 10.1109/MIC.2021.3103236.
20. D. Milojicic, P. Faraboschi, N. Dube, and D. Roweth, "Future of HPC: Diversifying heterogeneity," in *Proc. Des., Autom. Exhib. (DATE)*, 2021, pp. 276–281, doi: 10.23919/DATE51398.2021.9474063.

PEDRO BRUEL is a scientist at Hewlett Packard Labs, Milpitas, CA 95035 USA. Contact him at bruel@hpe.com.

SAI RAHUL CHALAMALASETTI is a senior scientist at Hewlett Packard Labs, Milpitas, CA 95035 USA. Contact him at sai-rahul.chalamalasetti@hpe.com.

ADITYA DHAKAL is a scientist at Hewlett Packard Labs, Milpitas, CA 95035 USA. Contact him at aditya.dhakal@hpe.com.

EITAN FRACHTENBERG is a master technologist at Hewlett Packard Labs, Milpitas, CA 95035 USA. Contact him at eitan.frachtenberg@hpe.com.

NINAD HOGADE is a scientist at Hewlett Packard Labs, Fort Collins, CO 80528 USA. Contact him at ninad.hogade@hpe.com.

ROLANDO PABLO HONG ENRIQUEZ is a senior scientist at Hewlett Packard Labs, Milpitas, CA 95035 UK. Contact him at rhong@hpe.com.

ALOK MISHRA is a scientist at Hewlett Packard Labs, Milpitas, CA 95035 USA. Contact him at alok.mishra@hpe.com.

DEJAN MILOJICIC is a Hewlett Packard Enterprise Fellow and vice president at Hewlett Packard Labs, Milpitas, CA 95035 USA. Contact him at dejan.milojicic@hpe.com.

PAVANA PRAKASH is a scientist at Hewlett Packard Labs, Milpitas, CA 95035 USA. Contact her at prakash@hpe.com.

GOURAV RATTIHALLI is a scientist at Hewlett Packard Labs, Milpitas, CA 95035 USA. Contact him at gourav.rattihalli@hpe.com.