GPU-NEST: Characterizing Energy Efficiency of Multi-GPU Inference Servers

Ali Jahanshahi, Hadi Zamani Sabzi, Chester Lau, and Daniel Wong

Abstract—Cloud inference systems have recently emerged as a solution to the ever-increasing integration of AI-powered applications into the smart devices around us. The wide adoption of GPUs in cloud inference systems has made power consumption a first-order constraint in multi-GPU systems. Thus, to achieve this goal, it is critical to have better insight into the power and performance behaviors of multi-GPU inference system. To this end, we propose GPU-NEST, an energy efficiency characterization methodology for multi-GPU inference systems. As case studies, we examined the challenges presented by, and implications of, multi-GPU scaling, inference scheduling, and non-GPU bottleneck on multi-GPU inference systems' energy efficiency. We found that inference scheduling in particular has great benefits in improving the energy efficiency of multi-GPU scheduling, by as much as 40 percent.

Index Terms—Multi-GPU, energy efficiency, inference server

1 Introduction

The ever-increasing demand for cloud-based inference solutions has spurred the development of APIs, runtimes, and hardware accelerators. For example, Google's Cloud Inference platform provides APIs to execute inference queries over large-scale, typed time-series datasets. NVIDIA released Triton Inference Server [1], an open-source cloud inference software stack solution that is optimized for NVIDIA GPUs. As another example, Qualcomm Cloud AI 100 inference accelerator and Intel Nervana NNP-I (Spring Hill) have released hardware accelerators for cloud inference solutions.

Despite the great benefits in lower power consumption and higher performance that application specific accelerators (ASIC) bring to the cloud inference services, GPU accelerators are still more popular in data centers due to their programmability. Integrating more GPUs into cloud systems has led to power hungry multi-GPU systems that gives rise to new power management challenges both at design and deployment time.

GPUs have been designed to perform most efficiently at peak utilization. Historically, this has been the common-case as many GPU-accelerated workloads (such as machine learning training, high-performance computing, and graphics) are designed to take advantage of as much computational power as possible. However, individual inference executions rarely utilize all the computational resources on the GPU [2] and would require concurrent processing of inference requests to maximize GPU utilization. Furthermore, the request-response nature of inference workloads varies during the day due to fluctuations in usage patterns, leading to potential underutilization. Under-utilization of GPUs pose challenges toward their energy efficiency, and can be exacerbated in multi-GPU inference systems without proper coordination and management.

Towards this end, the purpose of this paper is to characterize the energy efficiency of cloud multi-GPU inference servers. Energy efficiency metric is used since it captures both performance and power consumption of the system. Our experiments show that multi-GPU scaling, inference scheduling, and non-GPU bottleneck are the main contributors of multi-GPU system energy inefficiency. Specifically, this work makes the following contributions:

1) We present GPU-NEST, a methodology to characterize the energy efficiency of a multi-GPU inference system.
2) We use GPU-NEST to characterize the energy efficiency of several multi-GPU server configurations running the Triton Inference Server.
3) We explored the implications of inference models, inference scheduling, multi-GPU scaling, and non-GPU bottlenecks on multi-GPU inference system's energy efficiency.

To the best of our knowledge, our work is the first to characterize the energy efficiency of multi-GPU inference servers.

2 Multi-GPU Inference Server Overview

Fig. 1 shows an overview of an inference server. An inference server consists of an HTTP/Rest or gRPC *frontend* that receives inference requests (for example, in input image for object recognition) and sends back a response. These incoming requests goes to an *inference scheduler* that schedules a request to an inference backend. The *inference backend* executes the inference on a GPU. An *inference model* runs on an inference backend. There can be multiple inference models (and thus, inference backends) running concurrently at the same time, even on the same GPU. In addition, there can be different types of inference backends running simultaneously (i.e., Caffe, Tensorflow, TensorRT, etc.)

Inference Server. Many inference servers exist to serve pre-trained models, such as TensorFlow Serving, Multi Model Server [3], and Nvidia Triton inference server [1]. For the purpose of this paper, our goal is to explore the energy-efficiency characteristics of GPU-accelerated cloud inference servers. Towards this end, we decided to evaluate Nvidia's Triton inference server as it supports multi-GPU inference and is optimized for Nvidia GPUs.

In order to improve the inference throughput, Triton inference server by default uniformly distributes inference requests across multiple GPUs. Triton also support many optimizations, such as multiple dynamic batching policies, that are transparent to the client requesting inference.

Inference Backend. The Triton inference server supports pre-trianed models from different frameworks (TensorRT, TensorFlow GraphDef and SavedModel, ONNX, PyTorch, and Caffe2 NetDef) as the inference backend. The most GPU-optimized of these is Nvidia's TensorRT, which is an optimizer and runtime that speed-ups inference performance of machine learning applications. Pre-trained models are optimized by TensorRT, for example with layer fusion, and are then executed by the TensorRT inference engine.

3 Characterizing Energy Efficiency of Multi-GPU Inference Servers

In order to characterize energy efficiency of multi-GPU inference server systems, we present GPU-NEST, a methodology to instrument and measure power and performance characteristics of inference servers.

GPU-NEST Design. In this section, we detail our design of GPU-NEST for performing power-performance measurement of multi-GPU inference servers. Fig. 1 shows an overview of the design of GPU-NEST , which consists of two main components: a client-side load generator to generate inference requests, and server-side instrumentation of the inference server. For portability, GPU-NEST is containerized for easy deployment on NVIDIA GPU servers. GPU-NEST is modeled after SPECpower_ssj2008 [4] which is an industry-standard benchmark that evaluates the power and

- *The authors are with the University of California Riverside, Riverside, CA 92521. E-mail: {ajaha004, hzama001, clau009, danwong}@ucr.edu.*

Manuscript received 5 Aug. 2020; revised 3 Sept. 2020; accepted 7 Sept. 2020. Date of publication 14 Sept. 2020; date of current version 20 Oct. 2020.
(Corresponding author: Ali Jahanshahi.)
Digital Object Identifier no. 10.1109/LCA.2020.3023723

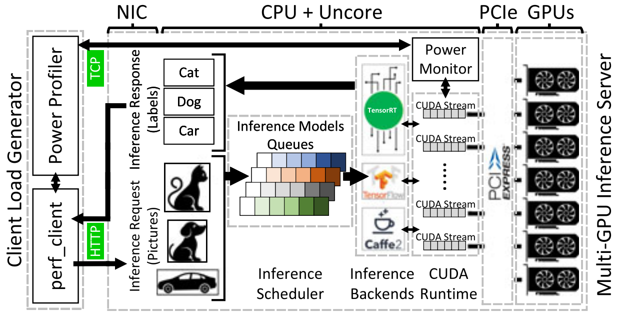


Fig. 1. GPU-NEST instruments GPU-accelerated inference server to measure energy efficiency characteristics.

performance characteristics of servers by measuring the performance (operations per second) and power (watts) consumption of servers at each 10 percent utilization interval.

Client-Side Inference Request Generator. Our client-side inference request generator is built on top of Nvidia's *perf_client* [5] tool and is able to generate inference requests at given rates (RPS). *perf_client* measures the throughput and latency of the generated requests over a time window and reports the steady-state metrics along with percentile confidence intervals. We present average latency in our results and observe similar trends with the tail latency.

To obtain the power and performance characteristics of GPU inference systems, GPU-NEST first identifies the maximum RPS that can be sustained by saturating the GPUs and observing the steady-state inference throughput. Next, *perf_client* profiles power-performance characteristics in 10 RPS *steps* from 0 to the maximum RPS. We find that each step only needs to run for ten seconds to obtain reliable and reproducible steady state statistics.

Server-Side Instrumentation. In order to measure the power of the GPUs on the server under test, we instrument a *Power Monitor* process, as shown in Fig. 1, that is synchronized with the client-side load generator. The power monitor starts recording the power of the server's GPUs by receiving a client-side `start_power_measure` command. The server-side Power Monitor stops power measurement and sends the average measured power of the GPUs to the client-side script when receiving `get_measured_power` command. The Power Monitor uses NVIDIA System Management Interface (*nvidia-smi*) for power measurement; it can also be easily extended to utilize external power monitoring tools if the GPU does not support power reporting.

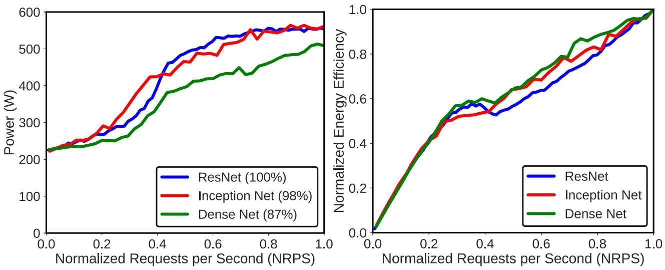
4 Evaluation

4.1 Experimental Setup

Table 1 shows the specifications of GPU servers used in our case studies. We select a variety of GPU-server configuration including an 8-GPU server (8x Telsa T4) and two 2-GPU servers (2x GTX 1070) with different non-GPU specs. As detailed in the previous section, we utilize the Triton inference server with the TensorRT inference backend.

TABLE 1
GPU Inference Server Platform Specifications

	8-GPU	2-GPU #1	2-GPU #2
CPU	Xeon Silver 4216	Core i7-6700K	Core i5-2500K
Sockets	2	1	1
Cores/socket	16	4	4
Threads/core	2	2	1
Memory	192 GB	16 GB	8 GB
PCIe speed	8 GT/s	8 GT/s	5 GT/s
GPUs	8x Tesla T4	2x GTX 1070	2x GTX 1070
GPU memory	16 GB (each)	8 GB (each)	8 GB (each)



(a) Power (b) Energy Efficiency

Fig. 2. (a) Peak power consumed varies by inference model executed and shows similar power curve shape. Max GPU util. for each model shown in parenthesis. (b) The energy efficiency curve of different inference models, with each model normalized w.r.t. itself, shows that different models will experience similar energy efficiency trends.

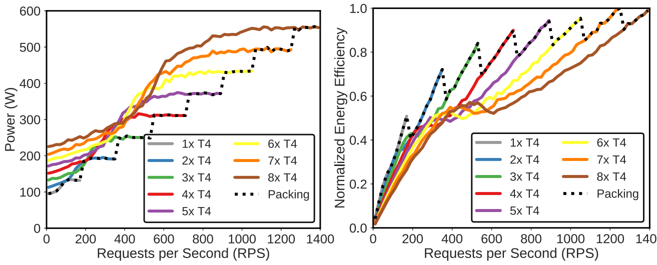
4.2 Case Studies

Implications of Inference Models. Inference throughput and latency are dependent on the type of inference model that is being executed. A larger model that requires more GPU resources would result in higher latency and lower throughput due to utilizing more hardware resources. To explore the impact of various inference models on energy efficiency, we run ResNet, Inception, and Dense Net on the 8-GPU server. As shown in Fig. 2a, both ResNet and Inception Net almost fully utilize the GPU (100 and 98 percent, respectively) and both consume ~557W at peak. For Dense Net, since it can only utilize 87 percent of the GPU due to lower computational requirements, Dense Net has a lower peak power at ~508W. While the peak power consumed varies by inference model, we observe the shape of the GPU's power profile remains similar and thus, the energy efficiency trends of different models are similar (Fig. 2b). In general, all GPUs running at lower utilization will also run at a lower energy efficiency. Based on these results, we use Resnet as the inference model under test as it maximizes GPU utilization.

Implications of Multi-GPU Scaling. Fig. 3 shows the case study where GPU-NEST measures the 8-GPU server's power and performance with 1 to 8 GPUs to explore how energy efficiency scales with GPU count.

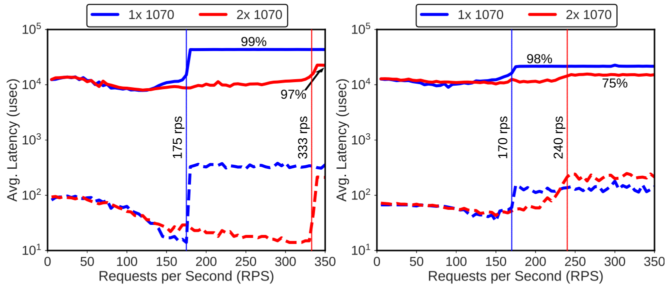
Observation 1. *GPU's active idle power affects overall energy efficiency of the system.* Idle Tesla T4 GPU consumes ~10W (we call this *Inactive idle power*). However, loading inference model(s) into a GPU increases its power by ~18W (we call this *Active idle power*). Thus, as Fig. 3a shows for $RPS = 0$, with a single GPU loaded with inference model(s), the overall 8-GPU system's idle power is nearly 98W (~10W for each idle GPU plus ~28W for the GPU loaded with model(s)). This systems' active idle power increases by ~18W per GPU as more GPUs are loaded with inference models. These results indicate that the GPUs perform idle power gating when the GPU is idle with no inference model loaded. This idle power can result in significant waste when the system's RPS load is low.

Observation 2. *No fixed GPU count achieves the best energy efficiency across different RPS.* There exist a trade-off in energy efficiency between active idle power, GPU count, and performance. For example, in Fig. 3b, at 1,000 RPS load, a 6-GPU configuration (6x T4) is more energy efficient than both 7-GPU and 8-GPU configurations. The 6-GPU configuration have all GPUs operating at near peak utilization, which corresponds to running at a higher energy efficiency (RPS / W). Meanwhile, the 7-GPU configuration has 7 GPUs running at a lower energy efficiency point. With the 7-GPU configuration at



(a) Power (b) Energy Efficiency

Fig. 3. Power consumption and normalized energy efficiency of an 8-GPU server (8x Tesla T4). Energy efficiency is measured as RPS / W and normalized to the 8x T4 case.



(a) 2-GPU #1 (b) 2-GPU #2

Fig. 4. Average inference scheduler *queue latency (lower dashed lines)* and GPU execution *inference latency (top solid lines)*. Vertical lines denote max sustainable RPS for a GPU count. Percentages denote GPU util. at max RPS.

this RPS, the performance gains do not outweigh the negatives of additional power added (yet not utilized) to the server and hence lower efficiency point. Therefore, having a fixed GPU count configuration does not provide the highest energy efficiency and there is a need for dynamically selecting GPU counts in inference servers.

While similar trends have been observed in other contexts, such as among servers in a cluster [6], [7], to the best of our knowledge, our work is the first to highlight this trend in multi-GPU servers. This opens up future work to see how applicable techniques design for cluster management can translate to managing GPUs within a server.

Implications of Inference Scheduling. Triton inference server provides uniform inference scheduling policy by default. To explore the impact of inference scheduling on energy efficiency, we implemented a *packing*-based scheduling policy. With packing scheduling, we send inference requests to a GPU until that GPU is fully utilized (without causing host-side queuing), before utilizing another GPU. Packing scheduling consolidates requests into a sub-set of active GPUs, while leaving another sub-set idle.

Observation 3. *The power and energy efficiency profile of the server with different GPU count follows the same trend using uniform inference scheduling policy.* As the number of GPUs scale, we observe that each *n*-GPU power curve is similar in shape to the 1-GPU case as GPU scales (Fig. 3a). This behaviour is similar for the energy efficiency profile (Fig. 3b). This pattern indicates that the inference requests are uniformly distributed across all GPUs in the Triton inference server which is energy efficiency agnostic.

Observation 4. *Inference scheduling policy can significantly improve energy efficiency without sacrificing performance.* The packing inference scheduler power and energy efficiency profile is shown in Fig. 3 with the black dotted lines. By packing responses to a sub-set of GPUs, this scheduling does not incur the overhead of increased active idle power, and gains the benefit of running GPUs at higher utilization, and thus, higher energy efficiency points. In Fig. 3b, we see that the packing scheduler effectively tracks the highest energy-efficiency configuration at a given RPS. The efficiency drops in the saw-tooth pattern is due to the increase in active idle power when a new GPU has a model loaded and running at a lower utilization. Packing scheduling can achieve 40 percent greater energy-efficiency levels than uniform scheduling on all 8 GPUs.

Implications of Non-GPU Specs. To explore the impact of non-GPU component configurations on GPU energy efficiency, we compare two servers running two Nvidia GTX1070 GPUs each. The 2-GPU #2 server has a less power CPU, less cores/threads, less memory, and lower PCIe bandwidth than those of 2-GPU #1 server. Fig. 4 shows the *scheduler queue* latency and *GPU inference execution* latency across a range of RPS. The maximum achievable RPS that does not cause exponential host-side queuing occurs when the GPU's are nearly fully utilized by the inference backend.

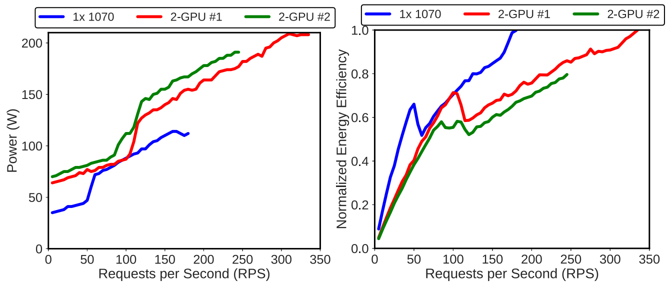
We note that GPU inference latency tends to be relatively stable and only spikes when it is fully utilized (>98 percent). After this point, the requests begin queuing on the host side (dotted lines). We observe a similar trend with the 8-GPU server. This further implicates that GPUs can tolerate packing scheduling to increase utilization without significant latency increases.

Observation 5. *Non-GPU bottlenecks can impact optimal multi-GPU configuration and limit energy efficiency.* When 1 GPU is running, both servers are able to fully utilize the GPU (98-99 percent) with nearly the same achieve max RPS of ~170 RPS. However, when both GPUs are utilized for Triton, non-GPU bottlenecks can limit the achievable RPS of the GPUs. Potential bottlenecks could be insufficient CPU processing power or PCIe communication bottlenecks. For example, 2-GPU #1 can achieve 333 RPS with 2 GPUs at 97 percent total GPU utilization, while 2-GPU #2 can only achieve 240 RPS with 2 GPUs at 75 percent total GPU utilization. Non-GPU bottlenecks can limit achievable GPU utilization, preventing the GPU from running at this highest efficiency range.

Fig. 5 shows the power and energy efficiency curve of a single GTX1070 (similar on both servers), and of both 2-GPU servers. What is notable is that for 2-GPU #1, since both GPUs can run at near maximum utilization, they both achieve the maximum energy efficiency that their max RPS. However, the 2-GPU #2 scenario cannot fully utilize the GPU due to non-GPU bottlenecks, limiting both GPUs to run at a medium utilization. *In fact, running both GPUs would significantly reduce energy efficiency by almost 30 percent.* Therefore, multi-GPU inference scheduling needs to take into account non-GPU bottlenecks in order to select the best multi-GPU configuration that maximize energy efficiency.

5 Related Work

Previous work on energy efficiency of GPUs has mostly focused on CPU-GPU heterogeneous systems by leveraging different



(a) Power (b) Energy Efficiency

Fig. 5. Power consumption and normalized energy efficiency of the 2-GPU servers. Energy efficiency is normalized to the 1x 1070 case.

approaches including DVFS- [8], [9], [10], [11], CPU-GPU workload division- or scheduling- [12], [13], architectural modification- [14], application- [15], and machine learning-based [16] techniques. The work of most relevance explores non-GPU bottlenecks of integrated CPU-GPU workloads with a focus on CPU energy efficiency impact [17] and bottlenecks of PCIe interconnects in multi-GPU systems with a focus on scalability [18]. However, none of these prior works explore the energy efficiency characteristics unique to multi-GPU servers or of inference servers.

Although several papers focused on characterizing or optimizing energy efficiency of single GPUs [8], [19], [20] and multi-chip module GPUs [21], and energy-efficient inter-GPU communication [22], *to the best of our knowledge, our work is the first to characterize energy efficiency of multi-GPU servers and of multi-GPU inference servers*.

6 Conclusion

We present GPU-NEST, a power benchmarking methodology for multi-GPU inference servers. Through various case studies on the Triton inference server, we explored the implications of inference models, multi-GPU scaling, inference scheduling, and non-GPU bottlenecks and the challenges they present to energy efficiency. In addition, we found that significant opportunities exist through inference scheduling and idle power management.

Acknowledgments

This work was supported in part by NSF Grant CCF-1815643 and the University of California, Riverside. The authors would like to thank the anonymous reviewers for their invaluable comments and suggestions.

References

[1] Nvidia, “NVIDIA triton inference server,” Accessed: Aug. 2020. [Online]. Available: https://docs.nvidia.com/deeplearning/triton-inference-server/

[2] P. Jain *et al.*, “Dynamic space-time scheduling for GPU inference,” 2018, *arXiv: 1901.00041*.

[3] Amazon Web Services - Labs, “Multi Model Server (MMS),” Accessed: Aug. 2020. [Online]. Available: github.com/awslabs/multi-model-server

[4] Standard performance evaluation corporation, “SPEC power,” Dec. 2007. [Online]. Available: https://www.spec.org/power_ssj2008/

[5] Nvidia, “Perf_client,” Accessed: Aug. 2020. [Online]. Available: https://docs.nvidia.com/deeplearning/triton-inference-server/master-user-guide/docs/perf_client.html

[6] D. Wong and M. Annavaram, “Implications of high energy proportional servers on cluster-wide energy proportionality,” in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit.*, 2014, pp. 142–153.

[7] D. Wong, “Peak efficiency aware scheduling for highly energy proportional servers,” in *Proc. 43rd Int. Symp. Comput. Archit.*, 2016, pp. 481–492.

[8] Q. Wang, P. Xu, Y. Zhang, and X. Chu, “EPPMiner: An extended benchmark suite for energy, power and performance characterization of heterogeneous architecture,” in *Proc. 8th Int. Conf. Future Energy Syst.*, 2017, pp. 23–33.

[9] T. Baruah, Y. Sun, S. Dong, D. Kaeli, and N. Rubin, “Airavat: Improving energy efficiency of heterogeneous applications,” in *Proc. Des. Autom. Test Europe Conf. Exhib.*, 2018, pp. 731–736.

[10] Z. Tang, Y. Wang, Q. Wang, and X. Chu, “The impact of GPU DVFS on the energy and performance of deep learning: An empirical study,” in *Proc. 10th Int. Conf. Future Energy Syst.*, 2019, pp. 315–325.

[11] J. Guerreiro, A. Ilic, N. Roma, and P. Tomás, “DVFS-aware application classification to improve GPGPUs energy efficiency,” *Parallel Comput.*, vol. 83, pp. 93–117, 2019.

[12] K. Siehl and X. Zhao, “Supporting energy-efficient computing on heterogeneous CPU-GPU architectures,” in *Proc. IEEE 5th Int. Conf. Future Internet Things Cloud*, 2017, pp. 134–141.

[13] E. Stafford, B. Pérez, J. L. Bosque, R. Beivide, and M. Valero, “To distribute or not to distribute: The question of load balancing for performance or energy,” in *Proc. Eur. Conf. Parallel Process.*, 2017, pp. 710–722.

[14] H. Asghari Esfeden, F. Khorasani, H. Jeon, D. Wong, and N. Abu-Ghazaleh, “CORF: Coalescing operand register file for GPUs,” in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, 2019, pp. 701–714.

[15] H. Zamani, Y. Liu, D. Tripathy, L. Bhuyan, and Z. Chen, “GreenMM: Energy efficient GPU matrix multiplication through undervolting,” in *Proc. ACM Int. Conf. Supercomput.*, 2019, pp. 308–318.

[16] R. Azimi, C. Jing, and S. Reda, “PowerCoord: A coordinated power capping controller for multi-CPU/GPU servers,” in *Proc. 9th Int. Green Sustain. Comput. Conf.*, 2018, pp. 1–9.

[17] A. Basu, J. L. Greathouse, G. Venkataramani, and J. Vesely, “Interference from GPU system service requests,” in *Proc. IEEE Int. Symp. Workload Characterization*, 2018, pp. 179–190.

[18] M. Chen, I. Chung, B. Abali, and P. Crumley, “Towards a single-host many-GPU system,” in *Proc. 30th Int. Symp. Comput. Archit. High Perform. Comput.*, 2018, pp. 140–147.

[19] J. Coplin and M. Burtscher, “Energy, power, and performance characterization of GPGPU benchmark programs,” in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, 2016, pp. 1190–1199.

[20] Y. Huang, B. Guo, and Y. Shen, “GPU energy optimization based on task balance scheduling,” *J. Syst. Archit.*, vol. 107, 2020, Art. no. 101808.

[21] A. Arunkumar, E. Bolotin, D. Nellans, and C. Wu, “Understanding the future of energy efficiency in multi-module GPUs,” in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, 2019, pp. 519–532.

[22] M. Khavari Tavana, Y. Sun, N. Bohm Agostini, and D. Kaeli, “Exploiting adaptive data compression to improve performance and energy-efficiency of compute workloads in multi-GPU systems,” in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2019, pp. 664–674.