# Can we optimize without specializing?
## A plea for generality in research systems

Aurojit Panda
*NYU*

## 1 Introduction

The systems research community spends a lot of its time optimizing systems to improve their performance and reduce resource requirements [6, 7, 13, 14, 16, 17, 20, 21]. We justify this research (to ourselves, our students and funding bodies) by arguing that these optimizations improve social utility: they make emerging technologies (e.g., machine learning or video streaming) more accessible to users by reducing costs, they reduce environmental impact by reducing the number of compute and network resources that need to be deployed, and allow new entrants into the market (encouraging further price reductions and improvements) by reducing overall deployment costs.

My thesis is that, in reality, many of the optimizations discussed in the systems' literature *do not* improve social utility, and instead they increase deployment costs, and make technology less accessible to users, deployers and developers. This leads to negative impacts on the environment and more importantly negative impacts on society, where it encourages the centralization of technology and eventually leads to an oligopoly.

The problems we are concerned about are not because optimizing systems is bad, but rather because one common approach to optimization which we refer to as *optimization by specialization*. This approach builds on the observation that specializing systems to particular hardware (*e.g.,* specific CPUs, GPUs, NICs, offloads and network interconnects), workloads (*e.g.,* request rates, value sizes, and popularity distributions), and deployment assumptions (*e.g.,* assumptions about who else might be sharing processing and communication resources) can yield performance and efficiency benefits for most systems. This approach lies at the core of many recent trends in systems (including trends that I participated in, benefited from, and continue to pursue), including kernel bypass networking [7, 14, 20], systems that build on RDMA or assume knowledge of communication latencies [13], disaggregated memory systems [1, 2, 12], systems that assume tensor cores or particular GPU features [8], and systems that assume new device interfaces and interconnects like CXL [12, 18].

My argument is not that we should not do this work, they have scientific value, and are crucial to understanding the design space and trade-offs that systems must make. Rather, my argument is that we should not blindly believe that deploying systems that have optimized by specialization is a net positive, and we should be spending some of our efforts generalizing these optimizations.

My concerns stem from the observation that deploying these systems requires adding new hardware, including servers, network switches and links, built using recently released (or sometimes pre-release) hardware, and then dedicating all (or most) of this newly deployed hardware to a single system that performs one very specific function. For example, training clusters for large language models, which are optimized using this approach require companies to buy new hardware, and deploy new network fabrics. Conversations about these clusters (both in private and in public forums such as the HotOS panel on sustainable systems) are dominated by concerns about the cost of this hardware, and supply chain concerns that increase the time required to build such a cluster. In many cases, the economic and environmental cost of these clusters outweighs the optimization benefits they provide. This is because the economic and environmental costs of these clusters must be paid upfront: deployers must pay the equipment costs, and we all must pay for the environmental costs associated with manufacturing [3] and powering [15] these clusters. Benefits on the other hand are small, and while they add up with each request, a substantial number of requests must be served before they outweigh the costs. Few, outside the largest providers, see sufficient requests within a system's lifetime (which is increasingly defined by time before a competitor announces a better equivalent) for the accumulated benefits to outweigh the costs.

This paper explores why we got here, *i.e.,* why are we optimizing by specializing? Then it discusses the concerns we see with this approach. Finally, we suggest some changes we can make as a community to address this concern.

## 2 Optimizing by Specializing

**Why do we optimize by specializing?** Optimizing systems by specializing them is appealing for several reasons. Superficially, it is both easy to empirically demonstrate the benefit of optimizations, and easy to motivate these works because they describe ways to use emerging technologies.

Beyond these superficial reasons, optimization by specialization gives us a way to improve the performance and efficiency of systems which have mature (and thus well optimized) implementations, without requiring algorithmic changes that can be either infeasible – either because the algorithm is close to being theoretically optimal or because we lack formal correctness requirements (as is the case with algorithms in machine learning training where correctness is often based on evaluation) – or require significant domain knowledge [4, 5, 10] and might not be approachable by many. By contrast, optimization by specialization requires using familiar techniques to identify bottlenecks, devise new abstractions, and demonstrate their efficacy.

**Why do our evaluations not reflect the environmental and economic costs of optimization by specialization?** At this point one might argue that the concerns I raised above (deployment costs and lack of sharing) are things that we can evaluate, and good systems papers likely already include evaluations for costs. In a recent paper [19], my colleagues and I discussed the problem with measuring end-to-end costs for systems that include accelerators. The problem I raise in this paper is harder because I am using a more expansive definition of end-to-end (in that paper we were not concerned about manufacturing costs) and we are talking about a wider range of systems. Therefore, I believe (though it is hard to cite a negative) that existing papers do not capture these costs, and furthermore, I am not convinced that we can enforce a meaningful evaluation standard that would capture these costs.

One might argue that commercial deployment (*e.g.*, the use of TPUs [9] or CDPUs [11] within Google, or the use of Megatron [13] at other companies) demonstrates that these costs are reasonable. However, we should be careful when using such arguments. Companies balance costs against benefits such as being first to market, showing technical leadership, etc. Therefore, adoption correlates with utility (does this help the company improve its perceived outlook) rather than cost.

**Why should we care about this problem?** As I noted previously, this paper is not arguing against research that optimizes by specialization. In fact, I plan to continue to working on this problem. My goal is to show that this line of work comes at a cost, and this cost is both social – we are designing systems that only a few of the richest companies in the world can deploy – and environmental. Therefore, we should find ways to encourage research that generalizes the lessons from this research and makes them more widely applicable. We discuss some ideas for this next.

## 3 What should we do?

Our communities research direction is often influenced by questions about how conferences and funding agencies evaluate research. Therefore, I think we should be more accepting of papers and research on systems that are slower but more robust than the state-of-the art. I use the term robust to describe systems that provide similar performance and efficiency across a variety of different deployments, including ones with different hardware, different workloads, or when sharing resources with different colocated applications.

At the same time, we should also begin to evaluate the systems we optimize by specialization to see what happens when deployment assumptions are violated, analyzing performance and efficiency when workload or sharing assumptions are broken. We should also do the same for cases when assumptions are made about the deployment hardware, though this requires additional work: a system might no longer function in the absence of some hardware feature. Perhaps, in this case a more robust design where the system falls back to using a more widely available legacy implementations, would both enable evaluation and be more desirable.

Finally, I hope that this short position paper leads to other solutions, better than the ones I have outlined above, and at least more detailed measurements and analysis that refute or support the thesis outlined here.

## References


[1] Emmanuel Amaro, Christopher Branner-Augmon, Zhihong Luo, Amy Ousterhout, Marcos K Aguilera, Aurojit Panda, Sylvia Ratnasamy, and Scott Shenker. Can far memory improve job throughput? In *EuroSys*, 2020.

[2] Emmanuel Amaro, Stephanie Wang, Aurojit Panda, and Marcos K Aguilera. Logical Memory Pools: Flexible and Local Disaggregated Memory. In *HotNets*, 2023.

[3] Daniel S. Berger, David Brooks, and et al. Reducing Embodied Carbon is Important. SIGARCH bloghttps://www.sigarch.org/reducing-embodied-carbon-is-important/, 2023.

[4] Edward Chen, Jinhao Zhu, Alex Ozdemir, Riad S Wahby, Fraser Brown, and Wenting Zheng. Silph: A framework for scalable and accurate generation of hybrid mpc protocols. In *S&P*, pages 848–863. IEEE, 2023.

[5] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 2022.

[6] Joshua Fried, Gohar Irfan Chaudhry, Enrique Saurez, Esha Choukse, Íñigo Goiri, Sameh Elnikety, Rodrigo Fonseca, and Adam Belay. Making kernel bypass practical for the cloud with Junction. In *NSDI*, 2024.

[7] Joshua Fried, Zhenyuan Ruan, Amy Ousterhout, and Adam Belay. Caladan: Mitigating interference at microsecond timescales. In *OSDI*, 2020.

[8] Daniel Y Fu, Hermann Kumbong, Eric Nguyen, and Christopher Ré. Flashfftconv: Efficient convolutions for long sequences with tensor cores. *arXiv preprint arXiv:2311.05908*, 2023.

[9] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, et al. TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *ISCA*, pages 1–14, 2023.

[10] Sheng-Chun Kao, Suvinay Subramanian, Gaurav Agrawal, Amir Yazdanbakhsh, and Tushar Krishna. Flat: An optimized dataflow for mitigating attention bottlenecks. In *ASPLOS*, 2023.

[11] Sagar Karandikar, Aniruddha N. Udipi, Junsun Choi, Joonho Whangbo, Jerry Zhao, Svilen Kanev, Edwin Lim, Jyrki Alakuijala, Vrishab Madduri, Yakun Sophia Shao, Borivoje Nikolic, Krste Asanovic, and Parthasarathy Ranganathan. Cdpu: Co-designing compression and decompression processing units for hyperscale systems. In *ISCA*, 2023.

[12] Huaicheng Li, Daniel S Berger, Lisa Hsu, Daniel Ernst, Pantea Zardoshti, Stanko Novakovic, Monish Shah, Samir Rajadnya, Scott Lee, Ishwar Agarwal, et al. Pond: CXL-based memory pooling systems for cloud platforms. In *ASPLOS*, 2023.

[13] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using Megatron-LM. In *SC*, 2021.

[14] Aurojit Panda, Sangjin Han, Keon Jang, Melvin Walls, Sylvia Ratnasamy, and Scott Shenker. NetBricks: Taking the V out of NFV. In *OSDI*, 2016.

[15] Brad Plumer and Nadja Popovich. A New Surge in Power Use Is Threatening U.S. Climate Goals. NY Times, https://www.nytimes.com/interactive/2024/03/13/climate/electric-power-climate-change.html, 2024.

[16] Deepti Raghavan, Shreya Ravi, Gina Yuan, Pratiksha Thaker, Sanjari Srivastava, Micah Murray, Pedro Henrique Penna, Amy Ousterhout, Philip Levis, Matei Zaharia, et al. Cornflakes: Zero-Copy Serialization for Microsecond-Scale Networking. In *SOSP*, 2023.

[17] Amanda Raybuck, Tim Stamler, Wei Zhang, Mattan Erez, and Simon Peter. HeMem: Scalable

Tiered Memory Management for Big Data Applications and real NVM. In *SOSP*, 2021.

[18] Hugo Sadok, Nirav Atre, Zhipeng Zhao, Daniel S Berger, James C Hoe, Aurojit Panda, Justine Sherry, and Ren Wang. Ensō: A Streaming Interface for NIC-Application Communication. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pages 1005–1025, 2023.

[19] Hugo Sadok, Aurojit Panda, and Justine Sherry. Of apples and oranges: Fair comparisons in heterogenous systems evaluation. In *HotNets*, pages 1–8, 2023.

[20] Irene Zhang, Amanda Raybuck, Pratyush Patel, Kirk Olynyk, Jacob Nelson, Omar S Navarro Leija, Ashlie Martinez, Jing Liu, Anna Kornfeld Simpson, Sujay Jayakar, et al. The Demikernel Datapath OS Architecture for Microsecond-Scale Datacenter Systems. In *SOSP*, 2021.

[21] Wen Zhang, Scott Shenker, and Irene Zhang. Persistent state machines for recoverable in-memory storage systems with NVRam. In *OSDI*, 2020.