# Of Apples and Oranges: Fair Comparisons in Heterogenous Systems Evaluation

Hugo Sadok
Carnegie Mellon University

Aurojit Panda
New York University

Justine Sherry
Carnegie Mellon University

## ABSTRACT

*Accelerators, such as GPUs, SmartNICs and FPGAs, are common components of research systems today. This paper focuses on the question of how to fairly compare these systems. This is challenging because it requires comparing systems that use different hardware, e.g., two systems that use two different types of accelerators, or comparing a system that uses an accelerator with one that does not. We argue that fair evaluation in this case requires reporting not just performance, but also the* cost *of competing systems. We discuss what cost metrics should be used, and propose general principles for incorporating cost in research evaluations.*

## CCS CONCEPTS

• **Hardware** → *Emerging technologies*; *Hardware accelerators*; • **Networks** → **Network performance evaluation**; • **Software and its engineering** → **Software performance**;

## KEYWORDS

Fair Comparison, Accelerator, Heterogenous Hardware

**ACM Reference Format:**
Hugo Sadok, Aurojit Panda, and Justine Sherry. 2023. Of Apples and Oranges: Fair Comparisons in Heterogenous Systems Evaluation. In *The 22nd ACM Workshop on Hot Topics in Networks (HotNets '23), November 28–29, 2023, Cambridge, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3626111.3628186

## 1 INTRODUCTION

The diminishing improvements in CPU performance over the last decade have contributed to a proliferation of different accelerators that promise better performance for specific types of applications. As a result, an increasing number of systems proposed in networking and systems conferences now incorporate devices such as SmartNICs [16, 20, 22, 25, 27, 30, 32, 33], FPGAs [4, 7, 10, 40, 42], and programmable switches [14, 15, 18, 24, 38, 39, 41]. With many of these systems reporting impressive improvements in performance metrics such as throughput and latency.


*HotNets'23, November 28-29, 2023, Cambridge, Massachusetts*

ACM ISBN 979-8-4007-0415-4/23/11.
https://doi.org/10.1145/3626111.3628186

Despite the performance improvements offered by these systems, it is often unclear if the proposed designs are a "win" over the existing ones. While many such systems are rigorously evaluated using an evaluation methodology that tries to be consistent with what the community has been doing for decades, this traditional methodology *breaks* when the underlying hardware changes. Traditionally, the way to compare two software systems is to run both in the same hardware device and compare their overall performance according to different performance metrics. But when two systems cannot be run on the same hardware device, looking only at performance metrics gives us an incomplete picture that does not account for the changes in cost between the two systems.

For instance, consider two typical claims found in papers that use heterogeneous hardware: "We have a system implemented in software; by pushing parts of it to a SmartNIC this system is now 2× faster." and "The system used to need 4 cores to achieve line rate, but by adding a programmable switch in front, it now only needs 2." The problem with these claims is that they only consider performance, even though the amount of available hardware resources changed. Could we achieve similar performance improvements by giving the original system more CPU cores? Why is using an accelerator any better than these additional cores or other accelerators that have already been proposed in the literature?

The main principle guiding this paper is that **systems evaluation with heterogeneous hardware should consider, and report, not only performance but also *cost*.** Our definition of cost is purposefully broad here. Just like performance can mean different things (e.g., throughput, latency, fairness), cost can refer to money, power, rack space, number of transistors, etc. In §3 we give a more precise definition of cost and propose principles that can be followed for picking good cost metrics depending on the systems being evaluated.

While the need to consider and report cost is intuitive and might even seem obvious in retrospect, many papers fail to properly consider cost in their evaluation. We believe that this is due to three main reasons:

**Lack of standard cost metrics:** While there is somewhat consensus on the main performance metrics that should be evaluated in a proposed systems (i.e., throughput and latency)—and even established benchmark methodologies for measuring these metrics [2]—the same is not true for cost. As a result, it is often unclear which cost metrics to pick, how to measure them, and which part of the system should

be included in the measurement.[1] Which cost metrics should one consider in an evaluation with heterogenous hardware?

**Cost asymmetry:** While performance reproducibility is a challenge in itself [17], at least we can agree on the value of different performance metrics given a fixed setup. But cost metrics are often "asymmetric," being perceived differently by different people and organizations, even when measured for the exact same setup. For example, Total Cost of Ownership (TCO) might be seen as a suitable cost metric, as it can technically be applied to any system and it is widely used in industry when purchasing systems. But TCO calculations can vary dramatically for different entities. Larger companies have access to lower prices due to the ability to bulk order. Moreover, the cost of rack space can be significantly different for a company operating in a big city vs. a company that operates in a remote rural area, where cost of land is cheap. How to pick cost metrics that do not suffer from this?

**Lack of cost information:** Another problem that often prevents researchers from incorporating cost in their evaluation is that cost information is often not public. Existing systems commonly do not report cost, making it hard to have a cost baseline when evaluating a proposed system relative to prior work. Moreover, companies often purposefully conceal cost information as a way to hamper competition. Datasheets can be vague or not publicly available, making it hard to infer cost for many hardware devices. How can evaluations measure cost with incomplete information?

In the rest of this paper, we propose principles and methodologies to avoid the above issues when evaluating systems with heterogenous hardware.

## 2 MOTIVATION

There has long been agreement on what a good evaluation should aim for: allowing a paper's readers to compare the system or design being proposed to other alternatives. While we have not always—arguably hardly ever—lived up to this ideal [2, 3, 23, 34], community standards have aimed to provide common metrics, measurement scenarios, and in some cases even benchmarks. For example, when evaluating network functions it is common to report both packets per second when using minimum sized packets and data rates when using a mixture of packets. Similarly, when building transactional databases, it is common to use one of the TPC benchmarks [35] and report transactions-per-second. Across all areas, it is also accepted practice to report what hardware and platform was used, and to compare systems running on the same hardware. This often requires that authors rerun previous systems on their setup.

Our focus in this paper is to ask how to compare systems that cannot run on the same hardware, as is the case for systems that propose new accelerators or propose novel uses of existing accelerators. There has been a steady growth in the number of systems and papers of this type [10, 22, 32, 38, 42], for good reasons: we are unlikely to see significant clock speed or transistor scaling in the future, energy and resource efficiency have become more important, and tools that allow researchers to prototype hardware accelerators have become accessible to the wider community. However, evaluating these systems can require comparing to older systems that run entirely on a CPU or comparing them to systems that use a different accelerator. Of course, we cannot require that these comparisons be run on the same hardware, but we should require "fair" comparisons, i.e., comparisons that meaningfully reveal trade-offs between systems.

Our core contention in this paper is that current evaluation approaches for accelerator-based systems are failing the community's fairness requirements. It is common to conclude from an evaluation that shows that system X, when run on 5 cores *and a SmartNIC*, has the same performance as system Y, which took 8 cores, that X is more "efficient" than Y. While we sometimes hide our discomfort, e.g., by arguing that SmartNICs should not be considered when evaluating efficiency, they are cheaper and, unlike cores, cannot be *resold to tenants*, but this is a weak argument that is unlikely to hold in another context. Similarly, it is also common to claim that a system X, that runs on 8 cores *and a SmartNIC*, is superior to another system Y, that only runs on 8 cores, since X achieves higher throughput than Y. This is despite system X using strictly more computational resources than Y.

The question we thus focus on is what is a fair way to compare systems that run on different hardware? Our observation is that when choosing what system to deploy or what design to adopt, what we really care about is not just whether the system has better *performance*, but also whether it improves their overall *utility*. A system's utility is dictated by both its *cost* and its *performance*, and thus a better system design is one that optimizes both cost (by lowering it) and performance (by improving it). Thus, our prescription when comparing systems that run on different hardware is to measure and report both cost and performance, and compare both. However, this simple prescription poses several challenges: how can one repeatably measure cost, after all, the price of hardware varies over time, organizations, and geographic location; and how does one compare normalize costs and performance across systems fairly. The rest of the paper addresses these questions.

A potential concern one may have with incorporating cost in an evaluation is that system prototypes are often not optimized for cost. Therefore, considering their cost in the evaluation may be too pessimistic when compared to a production system. We do not see this as a major roadblock. The fact that existing prototypes do not optimize for cost is, at least in part, an artifact of cost not being commonly reported in evaluations. As such, by encouraging the community to report cost, we are also encouraging better designs that are optimized for cost.

[1] Note that there are application-specific benchmarks that incorporate cost [28, 31, 36] but our goal is to define a set of principles that can be broadly applied to different systems.

Before moving on, we acknowledge we are not the first to consider incorporating cost metrics in an evaluation. Benchmarks such as JouleSort [28] (which measures performance and power for sorting), CloudSort [31] (performance and cloud computing cost for sorting), TPC-Pricing [36] (which provides a model for computing the cost of different TPC benchmarks), and computer architecture evaluations focused on performance-per-watt already incorporate cost into their measurements. The existence of these benchmarks further highlights the importance of considering cost when evaluating systems. This work complements these benchmarks by describing a methodology that can be broadly applied to a wide variety of accelerator-based systems.

# 3 PRINCIPLES FOR CHOOSING COST METRICS

A variety of metrics can be used to measure costs during evaluation, including capital expense (often dominated by hardware costs) [26], power utilization (watts) [11], administrator complexity [19], total cost of ownership (TCO), and rack-space [42]. **How should systems and network researchers choose between these metrics when evaluating systems?** The answer of course depends on what metrics we can feasibly measure for a system. But, not all feasibly measurable metrics are equally good, and we argue that researchers should choose metrics that have three properties: (a) they should be context-independent (§3.1); (b) they should be quantifiable (§3.2); and (c) they should measure end-to-end costs for all systems being compared (§3.3). Below, we explain these properties, suggesting some metrics that meet these criteria (§3.4).

Note we do not claim that metrics that do not fit these criteria are unimportant and should not be used: indeed, as we will see, TCO, a commonly used metrics for purchasing decisions, does not meet these criteria. Our message is different: it is hard to compare measurements from metrics that do not meet these criteria, and thus they are not good when reported in papers but might still be useful in other contexts.

## 3.1 Context-Independent Costs

One of the issues with cost metrics (that we discussed previously) is that some metrics can vary depending on deployment context, i.e., depending on when and where the measurement is taken, and by whom. Total Cost of Ownership (TCO) is an example of such a context-sensitive metric: the total cost of a system depends on where it is being deployed (which dictates energy costs, the cost of maintenance, etc.), who is deploying it (which dictates purchase cost), and can vary over time.[2] Consequently, even though TCO is arguably the most important cost metric when deciding what system to buy, it is a *poor metric* for research, since future researchers are unlikely to be able to compare against published TCOs.

[2]TCO's context dependency is probably at the root of the many heated debates we have seen about the value of specialized hardware. In particular when they involve people from companies that calculate TCO differently.

**Table 1: Examples of context dependent and independent cost metrics. Context-dependent cost metrics can be calculated differently depending on who is evaluating the system and when they are evaluating it. Context-independent cost metrics avoid this problem.**

| Cost Metric Type | Examples |
|---|---|
| Context Dependent | TCO ($), hardware price ($), carbon footprint ($CO_2e$) [12]. |
| Context Independent | Power (Watt), heat dissipation (BTU/h), silicon die area ($mm^2$), number of CPU cores, number of FPGA LUTs, memory usage (MB). |

To avoid this problem, we propose the following principle:

***Principle 1:*** *Cost metrics should be context-independent.*

A *context-independent* cost metric is a metric that yields identical costs for any two identical deployments of a system. We define identical deployments to be ones where the system runs on the same hardware, uses the same configuration, and processes the same workload. Thus, context-independence enables comparisons across space, organizations and time, because the value remains the same, regardless of *when* it is measured and by *whom*. We list common examples of context-dependent and independent cost metrics in Table 1.

Finally, we note that in conversations with collaborators and funders, we found that there was some discomfort with our stance on TCO: while people agreed that TCO is "context-dependent" they also pointed out that it is "the cost metric that companies care most about." This led us to ask, can one find context-independent equivalents to metrics such as TCO, which are inherently context-dependent? One approach is to release (with the paper) the pricing model used to compute the TCO, allowing others to compute TCO for their systems.

## 3.2 Quantifiable Costs

There are many cost metrics that capture important information but we currently have no good tools to measure. For example, sustainability has generated a lot of interest in the community [1, 8], and many have suggested using carbon footprint as a cost metric. However, there is no commonly agreed upon approach to measure this metric. Similarly, in software engineering, many are concerned about the additional programming complexity posed by platforms with accelerators, since accelerators add concurrency and require reasoning about interactions between devices with different consistency and coherence models. Programming complexity would thus appear to be a reasonable cost metric, however, there are no agreed upon metrics for measuring this. While task complexity is a common metric in the social sciences [5], there is wide-spread disagreement on how to measure it, and developing a repeatable metric for our field poses a significant challenge. Other important costs that are not easily quantifiable include manageability [9, 29] and

velocity [6, 21, 37]. We thus observe, that many important cost metrics cannot (yet!) be easily quantified or compared, which leads to our next principle:

***Principle 2:*** *Cost metrics should be quantifiable—measurable and comparable head-to-head.*

Of course, this principle does not mean that researchers should not discuss non-quantifiable cost metrics qualitatively. We merely suggest that they use this in *addition* to quantifiable cost metrics.[3] We also encourage the community to standardize new metrics that might enable us to compare today's unquantifiable costs using new, quantifiable, methods.

### 3.3 End-to-End Cost Coverage

Another concern when picking cost metrics is that many metrics fail to capture the "full picture." That is, they only measure cost for a few of the systems being compared, or they ignore portions of some systems.

For example, suppose we want to compare a system implemented using only a CPU with another that uses both an FPGA and a CPU. A cost metric such as number of FPGA lookup tables cannot be used here, as it cannot be measured for both systems. But even a metric such as number of CPU cores, which *can* be measured for both systems, is not suitable because it fails to cover all systems in the evaluation end-to-end, i.e., it does not account for the cost of the FPGA in one of the systems. The following principle describes this:

***Principle 3:*** *Cost metrics should cover all systems in the evaluation end-to-end.*

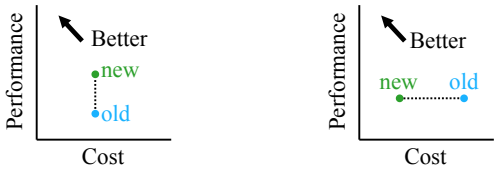
When we say that a cost metric has end-to-end coverage, it means that all components of the systems that are needed to produce the output are captured in the cost.

### 3.4 Practical Cost Metrics

Finally, having stated the three principles, we discuss how some commonly reported cost metrics fare on them:

- **Power draw (Watts):** Unsurprisingly, power meets all three of our requirements: it is context independent, measurable using a variety of tools, and can be added up and composed enabling end-to-end measurements.
- **Rack-space or space:** Rack-space does well in some aspects: it is quantifiable (it is just volume), and one can measure it end-to-end (space can be added up). However, it is not entirely context-independent. While there are standard rack-units, the number of devices that can be enclosed within depends on other concerns such as available power and cooling, and this makes it challenging. Thus, additional information must be provided when using rack-spaces as a context-independent cost metric.
- **Number of cores or number of LUTs:** These commonly reported cost metrics are, of course, context-independent and quantifiable. However, they are *not always* end-to-end: one cannot trivially add up cores or LUTs on different devices. Therefore, while number of cores (or LUTs) is a reasonable metric for a system that uses a single processing device, it does not work when comparing systems that use accelerators.

[3]There is an analogous problem for non-quantifiable *performance* metrics. For instance, reliability, while important, can be hard to quantify objectively in some contexts.




**(a) Improving performance.** **(b) Improving cost.**

**Figure 1: When systems operate in the same regime, they either have the same cost or the same performance. Comparison in such case is simple, as we only need to consider one dimension: performance or cost.**

We have already discussed other metrics such as TCO and carbon footprints earlier in the section, and shown that they do not meet our requirements. In the rest of the paper, we use power draw as our cost metric, but any cost metric that meets our three requirements can be substituted instead.

## 4 EVALUATION PRINCIPLES

A system's evaluation usually aims to achieve several goals, including:

- Motivating the system's design.
- Demonstrating that the system performs well when executing the targetted workloads in its targeted deployment conditions.
- Showing how the system's performance varies due to changes in workloads or deployment conditions.
- Evaluating trade-offs made when designing the system.

In all of these cases, it is often desirable to compare the system to others, using a variety of different types of metrics.

In this section, we describe a methodology for fairly comparing systems with heterogeneous hardware, by considering both performance and cost. Our hope is that such methodology can be applied systematically in different contexts where comparing multiple systems is desirable. For ease of exposition, our description is framed in terms of comparisons between an existing baseline system and a proposed system that is being evaluated. However, the approach generalizes when comparing larger numbers of systems, and under different circumstance. In what follows, we assume that the cost metric follows the principles in §3.

### 4.1 Operating Regime

The first step in comparing systems is to determine the performance and cost that these systems achieve. When systems under the same workload present the same cost or the same performance, we say that they *operate in the same regime.*

Comparing systems that operate in the same regime is simple because we can ignore the dimension that is the same for both systems. This is often the case when both systems

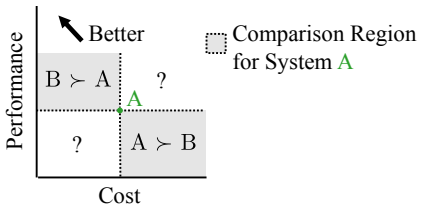



**Figure 2: Illustration of the comparison region for a proposed system A. When a baseline system B is in the comparison region of system A, it either dominates A ($B \succ A$) or is dominated by A ($A \succ B$). Alternatively, no relation can be inferred between the two systems.**

run on the same hardware. For example, if both systems run on an x86 CPU, saying that *"the proposed system improves throughput with a single core from 10 Gbps to 15 Gbps"* is a meaningful claim as both systems share the same cost (one CPU core). Similarly, saying that *"the proposed system reduces the number of cores required to saturate a 100 Gbps link from 8 to 4"* is also meaningful since both systems share the same performance (100 Gbps).

Figure 1 illustrates these two examples in the performance-cost space. Note how the analysis becomes unidimensional when both systems operate in the same regime, which lets us focus on the dimension that changes (either performance or cost). This notion is captured in the following principle:

> ***Principle 4:*** *When the proposed system and the baseline operate in the same regime, the analysis can be made unidimensional.*

One of the challenges in evaluating systems with heterogeneous hardware is that the baseline and the proposed systems often *do not* operate in the same regime. For instance, consider the example of a firewall that uses a SmartNIC, the system with the SmartNIC is likely to have higher performance but also higher cost than the baseline that does not need a SmartNIC. As we will see next, the analysis in such case must inevitably consider *both* performance and cost.

## 4.2 Scalable Systems and Metrics

When considering both performance and cost it helps to think of *Pareto dominance*. A design Pareto dominates another if it improves performance without sacrificing cost or it improves cost without sacrificing performance. The only conclusion one can draw from an evaluation that shows that the system Pareto dominates the baseline is that the system *both improves performance and reduces cost.*

When the proposed design does not Pareto dominate the baseline, or vice versa, we cannot make an objective claim of superiority. To represent all the possible designs where such a superiority claim is possible, we define what we call a *comparison region*. A comparison region of a given design comprises all possible designs which Pareto dominate or are Pareto dominated by such design. Figure 2 illustrates the comparison region for a proposed system A in the performance-cost space. Note how designs that are not Pareto dominated or dominated by A are outside its comparison region.

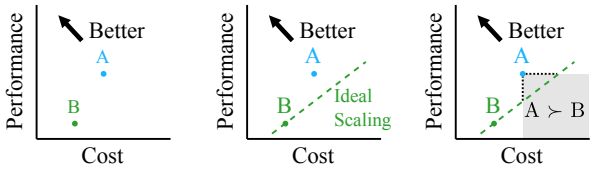



**Figure 3: Example of using ideal scalability to compare two designs: Accelerated (A) and Baseline (B). B is originally not in A's comparison region but we can generously estimate its performance and cost when operating at A's comparison region by assuming that it can scale linearly.**

When the baseline system can be horizontally scaled and when horizontally scaling the system also improves the performance metric, we should be able to scale the baseline to bring it to the comparison region of the proposed system. This is stated in the following principle:

> ***Principle 5:*** *Scalable baseline systems should be compared at the proposed system's comparison region.*

To make Principle 5 more concrete, consider again the example with the software firewall that is accelerated using a SmartNIC. Say the baseline system (using a regular NIC) achieves 10 Gbps and consumes 50 W of power when using a single CPU core while the proposed system (using a SmartNIC) achieves 20 Gbps and consumes 70 W of power. The baseline system is not in the proposed system's comparison region as it has worse performance but better cost.

To compare the two systems, we could scale the baseline so that it operates in the comparison region of the proposed system. In the example, we could give the baseline more CPU cores so that it can achieve a higher throughput. If the baseline running with two cores achieves 18 Gbps with 80 W, it is now in the comparison region of the proposed design. This lets us make an objective claim that the proposed system is better at this performance-cost target.

*4.2.1 Ideal Scalability.* One potential challenge in applying Principle 5 is that, even for scalable systems and metrics, scaling the baseline system may require substantial effort, or substantial more hardware, in order to scale horizontally. In such case, authors proposing the new system can significantly simplify the evaluation by assuming *ideal scalability*.

For example, consider again the accelerated firewall example. But now instead of using a SmartNIC we use a programmable switch to preprocess packets. The accelerated system can achieve 100 Gbps using all host cores and the programmable switch, while consuming 200 W. The baseline system provisioned to use all the host cores achieves 35 Gbps while consuming 100 W. The baseline system is again outside the comparison region of the proposed system but since we are already using all the host cores, we would need to provision multiple hosts in order to further scale the baseline.

Instead of trying to scale the baseline across multiple hosts, we can generously assume that the baseline system will scale linearly. This gives us a bound on how well the baseline system could do were we to use multiple hosts. In the example,

we can linearly scale the baseline until it matches the performance or the cost of the proposed system (e.g., 70 Gbps at 200 W or 100 Gbps at 286 W). Figure 3 illustrates this. Note how, by assuming ideal scalability, we can bring the baseline to the comparison region without having to actually provision it at a higher capacity. This is summarized in Principle 6:

***Principle 6:*** *When the baseline system and the performance metric are scalable, consider ideally scaling up the baseline to the proposed system's comparison region.*

There are a few potential pitfalls that one should be aware when using ideal scalability. The first is that one can only assume ideal scalability for the *baseline* and not for the *proposed* system, as assuming ideal scalability for the proposed system is no longer being generous to the baseline. Another potential pitfall is that one should be careful about the cost coverage when scaling. If, for instance, the baseline system originally does not use all CPU cores in the host, linearly scaling it using the cost of the entire server is no longer generous, as one could potentially extract more performance for the same cost by using more cores within the same host. The last potential pitfall is that not every system or metric is scalable. We address this last case in the following section.

### 4.3 Non-Scalable Systems and Metrics

While scalable baseline systems can often be provisioned to operate in the proposed system's comparison region, not every system can be scaled arbitrarily—or at all. This can be a problem in both directions: upscaling or downscaling. A proposed system may achieve a *performance* target that is not achievable by the baseline system—regardless of how much we try to scale the baseline. Alternatively, a proposed system may achieve a *cost* target that is not achievable by the baseline system—regardless of how much we try to downscale the baseline. In addition, some metrics do not scale when we scale the system, e.g., latency[4] and JFI [13].

There are two potential scenarios when dealing with non-scalable systems and metrics:

**Baseline in the comparison region:** When the baseline is already in the proposed system's comparison region, the systems are comparable and we can make an objective claim of superiority. For example, if the proposed system achieves 5 µs latency with 100 W power and the baseline achieves 10 µs latency with 300 W power, the proposed system is arguably superior as it improves both performance and cost.

**Baseline not in the comparison region:** When the baseline is *not* in the proposed system's comparison region, the two systems are fundamentally *incomparable*. For example, the proposed system may achieve 5 µs latency with 200 W power while the baseline system achieves 8 µs with 100 W power. In this case, one should still report both performance and cost for the proposed system. This lets readers decide if the system's operating regime fits their requirements. In addition, reporting performance and cost allows the system to be used more easily as a baseline when evaluating future systems, which may be comparable with the proposed one. Moreover, the authors of the proposed system should make a case for why achieving such a performance or cost target is desirable and provide evidence for why the additional cost or reduction in performance may be justifiable.

***Principle 7:*** *Non-scalable baseline systems are only comparable when they are originally in the proposed system's comparison region.*

[4]A system response latency for a fixed load might still improve with horizontal scalability, as systems often achieve lower latency at lower loads. However, there is usually a hard limit on how much one can improve latency in this way.

## 5 DISCUSSION AND CONCLUSION

This paper lays out a set of evaluation principles for systems that use accelerators. Our goal is to have these or similar principles influence how the community evaluates systems: we hope to drive the community to a place where authors adhere to these principles when evaluating their systems, and reviewers consider these principles when reviewing papers. Of course, we recognize that achieving this goal will require more than enumerating some principles. We thus hope to work with the community to develop good cost metrics, build tools and approaches for measuring them, and evaluating their utility when designing and comparing systems.

Two additional related questions raised by our principles are: (a) will industrial research groups be willing to report cost metrics; and (b) why are the cost metrics that fit our principles (i.e., metrics that are context-independent, quantifiable, and end-to-end) so different from TCO, which is used for purchasing decisions? Both questions share the same cause: cost and pricing information is often confidential, since it can reveal information about corporate relations (which affect discounts), salaries, contracts, and thus be a competitive disadvantage. While our hope is that the benefit of using cost information to influence system designs will alleviate some of these concerns, we are not certain of this.

Nevertheless, this is the right time for us as a community to consider how we want to evaluate and compare systems with accelerators: these systems are relatively new and our approach to evaluating and reasoning about them is still in its early stages, and thus amenable to change. This malleability is positive, since it allows research to influence what metrics are exposed by accelerator developers, and considered when building new systems, whether in academia or industry.

## ACKNOWLEDGMENTS

We thank the reviewers for their great feedback, Nirav Atre for his comments on an earlier draft, and Long Pham for suggesting the term "Context-Independent Cost."

This work was supported in part by Intel and VMware through the Intel/VMware Crossroads 3D-FPGA Academic Research Center, by a Google Faculty Research Award, and by ERDF through the COMPETE 2020 program as part of the project AIDA (POCI-01-0247-FEDER-045907).

## REFERENCES


[1] 2023. HotCarbon '23. https://hotcarbon.org/2023/index.html.

[2] S. Bradner and J. McQuaid. 1999. Benchmarking Methodology for Network Interconnect Devices. RFC 2544. https://doi.org/10.17487/RFC2544

[3] Aaron B. Brown, Anupam Chanda, Rik Farrow, Alexandra Fedorova, Petros Maniatis, and Michael L. Scott. 2005. The Many Faces of Systems Research: And How to Evaluate Them. In *Proceedings of the 10th Conference on Hot Topics in Operating Systems - Volume 10 (HotOS '05)*. USENIX Association, USA, 26.

[4] Marco Spaziani Brunella, Giacomo Belocchi, Marco Bonola, Salvatore Pontarelli, Giuseppe Siracusano, Giuseppe Bianchi, Aniello Cammarano, Alessandro Palumbo, Luca Petrucci, and Roberto Bifulco. 2020. hXDP: Efficient Software Packet Processing on FPGA NICs. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI '20)*. USENIX Association, 973–990.

[5] Donald J Campbell. 1988. Task complexity: A review and analysis. *Academy of management review* 13, 1 (1988), 40–52.

[6] Michael Dalton, David Schultz, Jacob Adriaens, Ahsan Arefin, Anshuman Gupta, Brian Fahs, Dima Rubinstein, Enrique Cauich Zermeno, Erik Rubow, James Alexander Docauer, Jesse Alpert, Jing Ai, Jon Olson, Kevin DeCabooter, Marc de Kruijf, Nan Hua, Nathan Lewis, Nikhil Kasinadhuni, Riccardo Crepaldi, Srinivas Krishnan, Subbaiah Venkata, Yossi Richter, Uday Naik, and Amin Vahdat. 2018. Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18)*. USENIX Association, Renton, WA, 373–387.

[7] Haggai Eran, Lior Zeno, Maroun Tork, Gabi Malka, and Mark Silberstein. 2019. NICA: An Infrastructure for Inline Acceleration of Network Applications. In *2019 USENIX Annual Technical Conference (ATC '19)*. USENIX Association, Renton, WA, 345–362.

[8] Babak Falsafi, Vijay Gadepally, Adam Belay, Carole-Jean Wu, and Andrew Chien. 2023. Panel on Sustainable Systems. (jun 2023). https://sigops.org/s/conferences/hotos/2023/program.html HotOS '23.

[9] Daniel Firestone. 2017. VFP: A Virtual Switch Platform for Host SDN in the Public Cloud. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)*. USENIX Association, Boston, MA, 315–328.

[10] Daniel Firestone, Andrew Putnam, Sambhrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian Caulfield, Eric Chung, Harish Kumar Chandrappa, Somesh Chaturmohta, Matt Humphrey, Jack Lavier, Norman Lam, Fengfen Liu, Kalin Ovtcharov, Jitu Padhye, Gautham Popuri, Shachar Raindel, Tejas Sapre, Mark Shaw, Gabriel Silva, Madhan Sivakumar, Nisheeth Srivastava, Anshuman Verma, Qasim Zuhair, Deepak Bansal, Doug Burger, Kushagra Vaid, David A. Maltz, and Albert Greenberg. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18)*. USENIX Association, Renton, WA, 51–66.

[11] Chang-Hong Hsu, Qingyuan Deng, Jason Mars, and Lingjia Tang. 2018. SmoothOperator: Reducing Power Fragmentation and Improving Power Utilization in Large-Scale Datacenters. In *ASPLOS*.

[12] ISO 14067:2018 2018. *Greenhouse gases – Carbon footprint of products – Requirements and guidelines for quantification.* Standard ISO 14067:2018. International Organization for Standardization, Geneva, CH. https://www.iso.org/standard/71206.html

[13] Rajendra K Jain, Dah-Ming W Chiu, and William R Hawe. 1984. *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems.* Technical Report DEC-TR-301. Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA.

[14] Xin Jin, Xiaozhou Li, Haoyu Zhang, Nate Foster, Jeongkeun Lee, Robert Soulé, Changhoon Kim, and Ion Stoica. 2018. NetChain: Scale-Free Sub-RTT Coordination. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18)*. USENIX Association, Renton, WA, 35–49.

[15] Daehyeok Kim, Zaoxing Liu, Yibo Zhu, Changhoon Kim, Jeongkeun Lee, Vyas Sekar, and Srinivasan Seshan. 2020. TEA: Enabling State-Intensive Network Functions on Programmable Switches. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 90–106.

[16] Jongyul Kim, Insu Jang, Waleed Reda, Jaeseong Im, Marco Canini, Dejan Kostić, Youngjin Kwon, Simon Peter, and Emmett Witchel. 2021. LineFS: Efficient SmartNIC Offload of a Distributed File System with Pipeline Parallelism. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP '21)*. Association for Computing Machinery, New York, NY, USA, 756–771.

[17] Shriram Krishnamurthi, Margo Seltzer, and Neeraja J. Yadwadkar. 2023. Panel on Future of Reproduction and Replication of Systems Research. (jun 2023). https://sigops.org/s/conferences/hotos/2023/program.html HotOS '23.

[18] Guanyu Li, Menghao Zhang, Cheng Guo, Han Bao, Mingwei Xu, Hongxin Hu, and Fenghua Li. 2022. IMap: Fast and Scalable In-Network Scanning with Programmable Switches. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI '22)*. USENIX Association, Renton, WA, 667–681.

[19] Hongqiang Harry Liu, Yibo Zhu, Jitu Padhye, Jiaxin Cao, Sri Tallapragada, Nuno P Lopes, Andrey Rybalchenko, Guohan Lu, and Lihua Yuan. 2017. Crystalnet: Faithfully emulating large production networks. In *SOSP*.

[20] Ming Liu, Tianyi Cui, Henry Schuh, Arvind Krishnamurthy, Simon Peter, and Karan Gupta. 2019. Offloading Distributed Applications onto SmartNICs Using IPipe. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 318–333.

[21] Michael Marty, Marc de Kruijf, Jacob Adriaens, Christopher Alfeld, Sean Bauer, Carlo Contavalli, Michael Dalton, Nandita Dukkipati, William C. Evans, Steve Gribble, Nicholas Kidd, Roman Kononov, Gautam Kumar, Carl Mauer, Emily Musick, Lena Olson, Erik Rubow, Michael Ryan, Kevin Springborn, Paul Turner, Valas Valancius, Xi Wang, and Amin Vahdat. 2019. Snap: A Microkernel Approach to Host Networking. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP '19)*. Association for Computing Machinery, New York, NY, USA, 399–413.

[22] YoungGyoun Moon, SeungEon Lee, Muhammad Asim Jamshed, and KyoungSoo Park. 2020. AccelTCP: Accelerating Network Applications with Stateful TCP Offloading. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*. USENIX Association, Santa Clara, CA, 77–92.

[23] John Ousterhout. 2018. Always Measure One Level Deeper. *Commun. ACM* 61, 7 (jun 2018), 74–83.

[24] Tian Pan, Nianbing Yu, Chenhao Jia, Jianwen Pi, Liang Xu, Yisong Qiao, Zhiguo Li, Kun Liu, Jie Lu, Jianyuan Lu, Enge Song, Jiao Zhang, Tao Huang, and Shunmin Zhu. 2021. Sailfish: Accelerating Cloud-Scale Multi-Tenant Multi-Service Gateways with Programmable Switches. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference (SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 194–206.

[25] Phitchaya Mangpo Phothilimthana, Ming Liu, Antoine Kaufmann, Simon Peter, Rastislav Bodik, and Thomas Anderson. 2018. Floem: A Programming System for NIC-Accelerated Network Applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*. USENIX Association, Carlsbad, CA, 663–679.

[26] Lucian Popa, Sylvia Ratnasamy, Gianluca Iannaccone, Arvind Krishnamurthy, and Ion Stoica. 2010. A cost comparison of datacenter network architectures. In *CoNEXT*.

[27] Yiming Qiu, Jiarong Xing, Kuo-Feng Hsu, Qiao Kang, Ming Liu, Srinivas Narayana, and Ang Chen. 2021. Automated SmartNIC Offloading Insights for Network Functions. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP '21)*. Association for Computing Machinery, New York, NY, USA, 772–787.

[28] Suzanne Rivoire, Mehul A. Shah, Parthasarathy Ranganathan, and Christos Kozyrakis. 2007. JouleSort: A Balanced Energy-Efficiency Benchmark. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*. Association for Computing Machinery, New York, NY, USA, 365–376.

[29] Hugo Sadok, Zhipeng Zhao, Valerie Choung, Nirav Atre, Daniel S. Berger, James C. Hoe, Aurojit Panda, and Justine Sherry. 2021. We Need Kernel Interposition over the Network Dataplane. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS '21)*. Association for Computing Machinery, New York, NY, USA, 152–158.

[30] Henry N. Schuh, Weihao Liang, Ming Liu, Jacob Nelson, and Arvind Krishnamurthy. 2021. Xenic: SmartNIC-accelerated Distributed Transactions. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP '21)*. Association for Computing Machinery, New York, NY, USA, 740–755.

[31] Mehul A. Shah, Chris Nyberg, and Naga Govindaraju. 2014. *CloudSort: A TCO Sort Benchmark*. Technical Report. The SortBenchmark Committee. https://sortbenchmark.org/2014_06_CloudSort_v_0_4.pdf.

[32] Rajath Shashidhara, Tim Stamler, Antoine Kaufmann, and Simon Peter. 2022. FlexTOE: Flexible TCP Offload with Fine-Grained Parallelism. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI '22)*. USENIX Association, Renton, WA, 87–102.

[33] Giuseppe Siracusano, Salvator Galea, Davide Sanvito, Mohammad Malekzadeh, Gianni Antichi, Paolo Costa, Hamed Haddadi, and Roberto Bifulco. 2022. Re-Architecting Traffic Analysis with Neural Network Interface Cards. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI '22)*. USENIX Association, Renton, WA, 513–533.

[34] Vasily Tarasov, Saumitra Bhanage, Erez Zadok, and Margo I Seltzer. 2011. Benchmarking File System Benchmarking: It *IS* Rocket Science.. In *HotOS*, Vol. 13. 1–5.

[35] TPC. 2023. TPC Benchmarks Overview. https://www.tpc.org/information/benchmarks5.asp.

[36] TPC. 2023. TPC-Pricing Homepage. https://www.tpc.org/pricing/.

[37] William Tu, Yi-Hung Wei, Gianni Antichi, and Ben Pfaff. 2021. Revisiting the Open VSwitch Dataplane Ten Years Later. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference (SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 245–257.

[38] Tao Wang, Xiangrui Yang, Gianni Antichi, Anirudh Sivaraman, and Aurojit Panda. 2022. Isolation Mechanisms for High-Speed Packet-Processing Pipelines. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI '22)*. USENIX Association, Renton, WA, 1289–1305.

[39] Zhuolong Yu, Yiwen Zhang, Vladimir Braverman, Mosharaf Chowdhury, and Xin Jin. 2020. NetLock: Fast, Centralized Lock Management Using Programmable Switches. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 126–138.

[40] Chaoliang Zeng, Layong Luo, Qingsong Ning, Yaodong Han, Yuhang Jiang, Ding Tang, Zilong Wang, Kai Chen, and Chuanxiong Guo. 2022. FAERY: An FPGA-accelerated Embedding-Based Retrieval System. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI '22)*. USENIX Association, Carlsbad, CA, 841–856.

[41] Kaiyuan Zhang, Danyang Zhuo, and Arvind Krishnamurthy. 2020. Gallium: Automated Software Middlebox Offloading to Programmable Switches. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 283–295.

[42] Zhipeng Zhao, Hugo Sadok, Nirav Atre, James C. Hoe, Vyas Sekar, and Justine Sherry. 2020. Achieving 100Gbps Intrusion Prevention on a Single Server. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI '20)*. USENIX Association, 1083–1100.