It Is Time to Address Network Power Proportionality

Lukas Röllin
ETH Zürich
Zürich, Switzerland
roellinl@ethz.ch

Romain Jacob
ETH Zürich
Zürich, Switzerland
jacobr@ethz.ch

Laurent Vanbever
ETH Zürich
Zürich, Switzerland
lvanbever@ethz.ch

Abstract

In recent years, networking hardware development has primarily focused on speed rather than power efficiency. By contrast, computing hardware has received a lot more attention given its dominant power footprint, especially in machine-learning (ML) data centers. With faster networks, we spend less time communicating and get more useful work out of the (increasingly expensive) computing hardware. But, the faster the network, the more time it idles and the worse its energy efficiency, which is magnified by the notorious lack of power proportionality of networking equipment.

In this paper, we analyze the network power footprint in a production ML cluster and find that it accounts for a still sizeable fraction of the total (12%) and that, by improving network power proportionality to match that of the compute, one could save close to 9% of the overall cluster energy demand. We argue that this potential is worth investigating and discuss opportunities and challenges to address power proportionality in networking hardware, which we invite the networking research community to tackle.

CCS Concepts

• **Hardware → Power estimation and optimization**; • **Networks**;

Keywords

Sustainable networking, Power proportionality, Machine learning infrastructure

ACM Reference Format:
Lukas Röllin, Romain Jacob, and Laurent Vanbever. 2025. It Is Time to Address Network Power Proportionality. In *The 24th ACM Workshop on Hot Topics in Networks (HotNets '25), November 17–18, 2025, College Park, MD, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3772356.3772382

HotNets '25, College Park, MD, USA

ACM ISBN 979-8-4007-2280-6/25/11
https://doi.org/10.1145/3772356.3772382

1 Introduction

Nowadays, the demand for faster network bandwidth is largely driven by data center applications and machine learning (ML) clusters in particular. To scale distributed ML training workloads, hyperscalers are arguing for "an increase in network bandwidth towards 400Gbps and higher" [14]. The idea is to complete communication faster, such that GPUs can spend more time computing and less idling.

The flipside of a faster network is that the network is then mostly idling, which leads to low energy efficiency, especially since networking hardware has notoriously poor power proportionality; *i.e.*, the power draw remains roughly constant when the network is idle [9, 15, 23, 33]. This is usually seen as acceptable since the power demand of the compute infrastructure largely dominates the network demand. However, as energy is becoming an increasingly critical resource, it is worth revisiting that assumption.

Thus, in this paper, we study the impact of power proportionality on a cluster's performance in terms of iteration time, power, and energy efficiency. The analysis of a production cluster and workload reveals that the network accounts for a not-so-small 12% of the cluster's energy demand, consumed with an appallingly low efficiency of 11%. Improving network power proportionality to 50%—still far worse than the proportionality of compute [3]—could save around 5% of the total cluster power. Increasing the proportionality to the same level as compute brings the savings to almost 9%. Note that, as computing is particularly power-intensive, ML clusters represent a sort of worst-case application to argue for network power proportionality. But even then, given the scale of power demand in ML clusters, 5–10% represents sizable energy and cost savings that, alone, arguably justify giving network power proportionality some consideration.

The historical approach for this is "link sleeping," which was studied [19] then implemented in the Energy Efficient Ethernet (EEE) standard (802.3az [8]) in the 2010's. But with the increase in network bandwidths and the adoption of optical technology, those techniques became effectively obsolete. In this paper, we revisit network power proportionality by reviewing the approaches used to make computing hardware power proportional today and discussing the opportunities and challenges to transpose them to networking hardware.

§ 4.1 Expose more power knobs in network operating systems and fix power gating;
§ 4.2 Extend datacenter network topology with optical circuit switches to tailor the topology to the workload;
§ 4.3 Clock ASIC pipelines independently to enable efficient rate adaptation;
§ 4.4 Extend the router design with a circuit switch to enable aggregation of traffic in some pipelines and turn others off in low-load conditions.

Finally, in § 4.5, we reflect on the potential opportunities offered by a complete ASIC redesign with power proportionality as the primary objective, which could include much smaller pipelines or co-packaged optics.

2 Modeling Approach

We discuss how the network bandwidth and power proportionality affect the total cluster power draw and workload completion time, which requires models of the workload, power, and network topology. We present those models below (§ 2.2 to 2.4) after introducing the production cluster we consider as a baseline (§ 2.1).

2.1 Baseline Cluster and Workload

As a baseline, we use one pod of the production cluster and workload described in a paper from Alibaba [27], featuring:

- 15k hosts, running on Nvidia H100 GPUs, 8 per server,
- a 400 G interface per GPU,
- a fat tree topology using 51.2 Tbps switches,
- a workload with a 10% communication ratio (§ 2.2).

2.2 Workload Model

We assume that a training workload is executed as a sequence of computation and communication phases. One computation and one communication phase make up one iteration. We assume no overlap between the phases; i.e., during computation, GPUs are working at full speed while the network is idle, and vice versa.[1] The *communication ratio* is the time of the communication phase divided by the iteration time. In addition, we assume that the total workload is constant as we scale the cluster; *i.e.*, we neglect the overhead from compute distribution or communication latency. Finally, we assume the workload execution time to scale linearly with the hardware resources; *i.e.*, a cluster with 2× GPUs completes a computation phase twice as fast.

Figure 1 summarizes our model for how the workload execution time scales with available hardware resources.

[1]While some training methods violate that assumption [10], it is compatible with recent papers describing large-scale ML training clusters [27] and in line with the traffic patterns reported in the CASSINI paper [28].

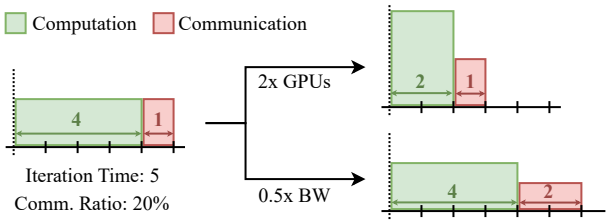


Figure 1: Our model linearly scales the workload execution time with the amount of resources available.

Table 1: Power values for GPUs and switches.

Device	Power (W)
Nvidia H100 NVL	400 [21]
51.2 Tbps Switch	750 [27]

Table 2: Power values for the network components.

Bandwidth (Gbps)		100	200	400	800	1600
NIC	Power (W)	8.6	16.7	25.4	38.6*	58.8*
Transceiver	Power (W)	4	6.5	10	16.5	27.2*

* extrapolated values

2.3 Power Model

Our model assumes that the hardware resources are either idle or operating at full speed (§ 2.2), mapping to two power states: idle and max. *Power proportionality* is defined as

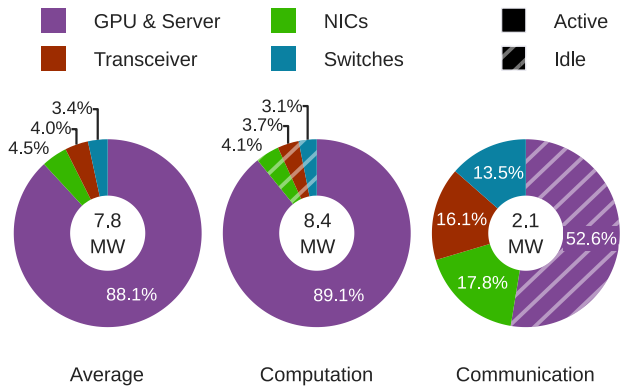
$$\text{power proportionality} = \frac{\text{max power - idle power}}{\text{max power}} \quad (1)$$

2.3.1 Compute power model. The Nvidia H100 GPU is rated at a max power of 400 W (Table 1). Each server hosts 8 GPUs and we assume it draws approximately 800 W for its other components (CPUs, RAM, storage, and fans), leading to a max power of 500 W per GPU. Modern servers have power proportionality reported at around 85% [4], which we use in our power model. This results in a GPU idle power of 75 W.

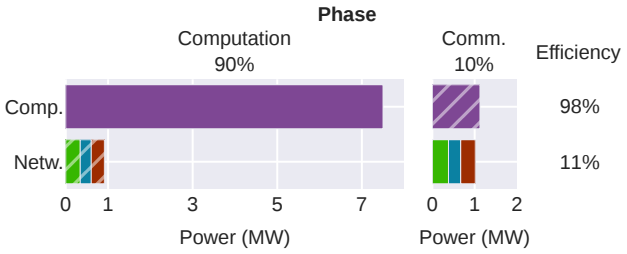
2.3.2 Network Power Model. For the network, we consider switches, network interface cards (NICs), and transceivers.

We use the number reported by Alibaba [27] as max power for switches (Table 1). We use datasheet power numbers for the NICs (NVIDIA's ConnectX-7 cards [20]) and the transceivers [13]. For the interface speeds where there are no NICs available, we linearly extrapolated from the closest available one (Table 2). We consider electrical transceivers between the GPUs and the top-of-rack switch (≈ 0 W) and short-range (< 2 km) optical ones between switches.

Networking hardware has notoriously low power proportionality, estimated between 5 and 20% [9, 15, 23, 33]. We model the network with a baseline proportionality of 10%.



(a) The network represents a small part of the power draw during computation, but almost half during communication.



(b) While the compute hardware is able to reduce its power draw when idle, the network's is almost constant.

Figure 2: Power footprint of the baseline cluster (§ 2.1)

2.4 Network Model

Increasing the number of GPUs or changing the bandwidth per GPU influences the size of the network. To perform our analysis, we need the number of switches required to connect all the GPUs. To calculate this, we use the formula from [26] for fat-tree topologies and interpolate if the number of hosts is between what a certain number of stages could support.

3 Impact of Power Proportionality

In this section, we quantify the power draw that is attributed to the compute infrastructure and the network within the baseline ML cluster, to show the ratio of the overall power that is used by the network. We evaluate the impact that more power-proportional network devices would have on the cluster's power draw. Since power has become a major factor for new ML training clusters [5], we also evaluate how much performance is left on the table due to network devices taking away power that could be allocated to compute.[2]

[2]The evaluation code is available here: https://github.com/nsg-ethz/network_powerprop_hotnets25

Table 3: Power savings of the total ML cluster compared to today's network with 10% power proportionality.

Bandwidth (per GPU)	Power Proportionality 10%	20%	50%	85%	100%
100G	0.0 %	0.3 %	1.2 %	2.3 %	2.7 %
200G	0.0 %	0.6 %	2.5 %	4.8 %	5.7 %
400G	0.0 %	1.2 %	**4.7%**	**8.8** %	10.6 %
800G	0.0 %	2.2 %	8.7 %	16.4 %	19.7 %
1600G	0.0 %	3.9 %	15.6 %	29.3 %	35.1 %

3.1 Compute vs Network Efficiency

First, we quantify the power footprint of the compute vs the network hardware for our baseline cluster (§ 2.1). Unsurprisingly, compute dominates (Fig. 2). However, it is interesting to observe how the power evolves during the computation and communication phases, both in relative (Fig. 2a) and absolute (Fig. 2b) terms.

During the computation phase, compute represents 88% of the total. The split with network power is more even during the communication phase, close to 50/50. This is because the computing hardware is able to reduce its power when idle, down to 15% of its maximum power (§ 2.3). Conversely, the network power, though smaller in magnitude than the compute power (12% of the total on average), remains almost constant when the network is idle. As the network is idling 90% of the time, the energy efficiency of the network infrastructure reaches an appallingly low value of 11%.

The rest of this section first considers the power savings that would result from better network power proportionality (§ 3.2) and then quantifies the performance speedup that those savings could enable by freeing power budget to add more GPUs to the cluster (§ 3.3).

3.2 Network Power Proportionality

As discussed in § 3.1, the network power footprint is relatively small compared to the compute's, so one may dismiss the lack of network power proportionality or energy efficiency as negligible.

In this section, we investigate that assumption by quantifying the potential gains from improving the network power proportionality for our baseline cluster and assuming different network bandwidth. Table 3 shows the relative gains compared to a 10% network power proportionality.

Result: We find that, on the baseline cluster, we would save around 5% of the overall power with a network that is 50% power proportional. As expected, the savings increase by going to higher power proportionalities. If the network proportionality were 85%, *i.e.*, on par with the compute's, the savings would reach 9%.

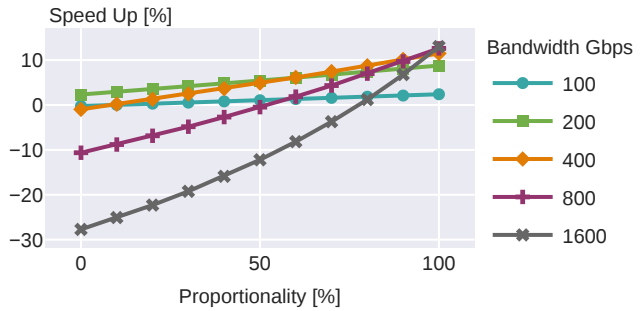


Figure 3: The most efficient network bandwidth depends on the network's power proportionality.

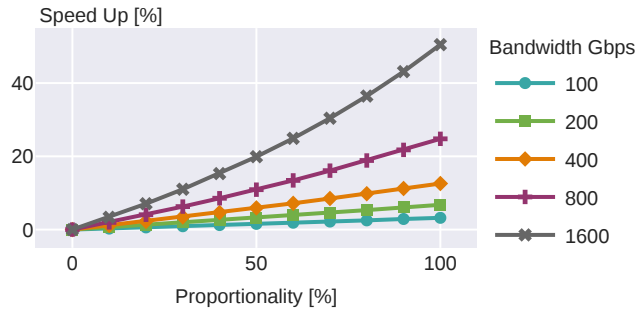


Figure 4: The increased power draw of higher network bandwidth makes proportionality more important.

While one may dismiss a few percentage points, at this scale, those translate to significant savings both in power and operating costs. For the 400 G case, 5% power savings convert to an average power draw reduction of 365 kW. Taking the average electricity price for the commercial sector in the US (13 cents per kWh, [11]) results in $416k/year saved on the electricity bill. In addition, the power consumption of the cooling infrastructure is estimated in [35] to be around 30% of the cluster power, adding another $125k/year in savings.

There would be other benefits, such as the flattening of the peak power demand, which reduces the strain on the power delivery system, though those are harder to quantify.

3.3 Performance Speedup

Nowadays, data centers are starting to become limited by the amount of power that is available [5]. Thus, every Watt that would be saved on the network side via better proportionality could potentially be used to put more GPUs in the same cluster. In this section, we aim to quantify the corresponding performance speedup that could be obtained while keeping a fixed power budget.

We consider two different scenarios. First, we consider a fixed workload, which implies that the length of the communication phase (and thus, the iteration time) changes with the network bandwidth. Second, we consider a fixed communication ratio, which implies that the workload scales with the network speed.

Fixed Workload: For a fixed workload, larger bandwidth implies decreased communication time, which influences the iteration time as well as the average power, since we now spend even more time in the computation phase. However, we also have to reduce the number of GPUs since the network devices consume more of the power budget, which in turn results in a longer computation phase. The best iteration time is obtained by balancing the power budget between the network and compute.

Figure 3 shows the iteration time for different network power proportionalities and network bandwidths. All of the speed-up numbers are in reference to the baseline scenario. We find that, in this scenario, lower network bandwidth is faster overall if the network power proportionality is poor. It is somewhat counterintuitive that a lower network bandwidth leads to faster iteration times: reducing the network bandwidth makes the GPUs idle for longer while the compute power dominates. The reason is twofold:

(1) Due to its low proportionality, the network power draw remains approximately the same when idling. Increasing the network bandwidth means increasing the network idle power, thus wasting even more power on the network side.
(2) At the same time, by increasing the network bandwidth, we reduce the time spent on communication, which means the network idles for a larger part of the iteration, making the network even less efficient.

Better power proportionality improves the iteration time for all bandwidth speeds as it frees some power budget for more GPUs. However, even at 50% proportionality, a 200 Gbps network is still faster than a 400 Gbps one. 800 and 1600 Gbps speeds become the best alternatives only at very high proportionality values ($\geq$ 90%). However, this is assuming a communication ratio that shrinks to 5% or 2.5%, which feels far from realistic training workloads. This leads us to our second evaluation scenario.

Fixed Communication Ratio: In this scenario, we look at a fixed communication ratio of 10%, or in other words, the communication workload increases with the network bandwidth. Figure 4 shows the iteration time speedup compared to a network with zero power proportionality. Looking at this scenario, we see that the higher the bandwidth, the bigger the performance gain. This is intuitive as the network takes up a bigger portion of the overall power and, therefore, benefits more from higher power proportionality. The magnitude of the speedup is worth noting: *e.g.*, a network power proportionality of 50% on a 800 Gbps network would enable a 10% speedup.

3.4 Discussion

Whether we look at it from the perspective of power savings (§ 3.2) or performance speedups (§ 3.3), we established that better power proportionality could deliver sizable benefits in ML training. For power proportionality improvements to yield sizeable benefits, there must be underutilization in the network. In § 3, we assumed the network is idle while the computation happens, but other forms of underutilization can be exploited. If we relax our assumption and allow computation and communication to overlap during training, as is done in other training schemes, there is still underutilization, as not all paths in the network are used all the time, especially in full bisection bandwidth networks. There is potential in other network contexts too; *e.g.*, in ISP networks, the benefits from power proportionality are even more direct since it is all network and no compute. In ISPs, underutilization is unavoidable since customers expect capacity to be there, but will not be using it 24/7. Unlike for ML training, traffic is less predictable, and links are more likely to be underutilized rather than completely unused; this is a different kind of underutilization that offers distinctive challenges and opportunities for power savings with better proportionality.

In the next section, we discuss several approaches for proportionality improvements, which may benefit underutilized networks in general—not solely focused on Machine Learning training clusters.

4 Achieving Power Proportionality

Networks are expected to provide any-to-any connectivity at increasingly higher speeds and lower delays. Those objectives conflict with the implementation of power-saving features, such as opportunistically turning components off. By contrast with computing chips or embedded systems, those features have been generally deprioritized in wired networking hardware, leading to poor power proportionality.

In 2008, academics theorized how to implement power savings at the link level [19]. These promising principles were then implemented into the Energy Efficient Ethernet (EEE) standard (802.3az [8]). However, as speeds increased and optical links became common, EEE lost its appeal.

In § 3, we showed how beneficial better network power proportionality could be, even in compute-dominated applications such as ML clusters. Thus, we argue that it is time to revisit this line of research and address network power proportionality. We take inspiration from the methods used to optimize the power of computing hardware. Those can be categorized into static optimization, *i.e.*, adjusting the hardware state to reduce its power draw at zero load, and dynamic optimization, *i.e.*, better scaling of the power demand with the load. We highlight two principles for each category and discuss how they could be transposed to networking.

4.1 Static Opt. #1: Exposing Power Knobs

Inspiration from Compute: Often enough, the hardware provides more features than needed for a given application. The simplest way to reduce static power is to include power gating mechanisms that enable turning off unused components, such as PCIe slots or memory banks. The same principle can be applied to OS features or background tasks.

Network: Hardware "bloat" exists in networking hardware as well, and, in many cases, components remain on and draw power. *E.g.*, a router may contain enough memory to store all the BGP routing information, but only needs to store a small part if deployed in a network with route reflectors. On the routers from major manufacturers we could study, we found no direct way to power off unused components. The user is blind and bound to the power knobs exposed by the closed-source OS, which are few. Blog posts from Juniper [34] show that some power gating in the pipelines is possible (and highly efficient). But even if the hardware supports power gating, the knobs must be exposed to users.

Some knobs are already in the user's control: *e.g.*, turning ports on and off. However, recent works highlighted that even though the ports are off in software, they may still be powered on in hardware [15, 24]. Similar power issues have been reported by hardware vendors, too [12]. At present, there is no evidence suggesting that those power inefficiencies are inherent hardware limitations; those seem to be software bugs, which can (and should) be fixed.

Challenges: Power gating requires hardware support and comes with some overhead. What components should be power-gated? Which knobs should be exposed to the user, and which should be dialed automatically? For example, if the switch is only configured for L2 forwarding, it could automatically turn off all L3 functionality. Assuming more knobs get exposed, the users would need a good understanding of the inner workings of the device to know which components can be turned off for their application; this conflicts with the current practice of hardware vendors disclosing little about how the hardware is built. This could be addressed with the networking equivalent of C-states for CPUs, *i.e.*, a list of pre-defined low-power modes that the switch supports without exposing the underlying hardware details.

4.2 Static Opt. #2: Scheduling Network Jobs

Inspiration from Compute: In compute clusters, a job scheduler is used to assign workloads to specific servers and can be used to concentrate the workload on as few servers as possible. This frees up the other servers to be run in low-power modes or, ideally, be turned off. There are many examples of using job schedulers to concentrate workloads [6, 17, 30, 31].

Network: Applied to networking, this approach would concentrate the network traffic on as few devices as possible. ML training is an ideal application for this since traffic patterns are very predictable and stable over time. We argue that, in such cases, a network topology that can handle any traffic pattern, *e.g.*, a fat tree, is not always needed. Techniques such as oversubscription help, but do not provide workload-specific flexibility.

Instead of keeping the complete topology active at all times, one could integrate optical circuit switches (OCS) to match the topology to the traffic pattern while minimizing the network switches that must remain on. The idea of optical network reconfiguration is not new, but previous works, such as Rotornet [18] or Sirius [1], are practically limited by their need for reconfiguration time within a few ns. But if we consider ML training jobs, which may last for days and would require only one reconfiguration when the job starts, off-the-shelf OCSs that feature reconfiguration times of a few tens of ms would be sufficient.

Challenges: Where should OCSs be added? It is trivial to optimize the network topology by placing an OCS in front of every switch, but this is a large overhead. There are interesting questions in optimizing the number and placement of OCS elements in the cluster topology. This may also leverage the known traffic patterns in ML workloads. The reconfiguration should ideally only happen when a new job arrives, but it is also necessary if the jobs change their network pattern. Turning on network devices takes a while, so it makes sense to keep some devices in standby. This provides an interesting trade-off question between energy savings and reaction time.

4.3 Dynamic Opt. #1: Rate Adaptation

Inspiration from Compute: For a given static hardware configuration, one may optimize the dynamic power, *i.e.*, how the power scales with the workload. The classic approach in CPUs is dynamic voltage and frequency scaling (DVFS), which would, *e.g.*, reduce the clock speed of some cores to match the current load [2, 16, 22].

Network: DVFS can be applied to the routers' control plane, but as it represents a small part of the total footprint, the benefits would be limited. The principle may also be applied to the data plane, though, and is then usually called *rate adaptation* [19]. The idea is to save energy by reducing the speed of the packet pipelines to match the current load. This can already be done on some routers today, but only globally: All pipelines will be controlled jointly by the ASIC's frequency. Another possibility is to configure an interface to a lower speed, *e.g.*, set a 100G-capable interface at 10G, which may save power by enabling turning off some of the interface's SerDes lines. This has been observed [15], but down-rating is not widely supported, and savings are limited—supposedly because few components are powered off.

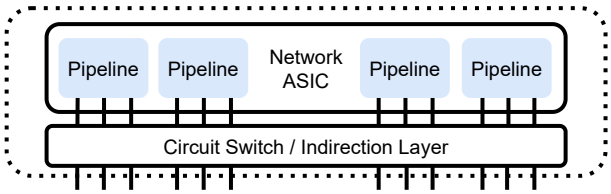


Figure 5: Adding a circuit switch allows for flexible assignment of ports to pipelines on the network ASIC.

On the plus side, rate adaptation techniques would also apply to transceivers and network interface cards (NICs), which make up a large part of the overall network power (Fig. 2).

Challenges: To make rate adaptation efficient, each pipeline should run at its own frequency and dynamically adapt it to the load. However, that suggests a more complex clock tree in the ASIC design, which is a substantial change.

Moreover, to make rate adaptation really efficient, the ASIC frequency scaling should operate in conjunction with the power-gating of other elements in the pipeline, such as memory blocks or SerDes lines. To realize the vision of dynamically scaling to the traffic load, those adaptations should be handled automatically by the operating system.

4.4 Dynamic Opt. #2: Turning off Pipelines

Rate adaptation keeps most components powered on. To get larger savings, we must turn entire pipelines off.

Inspiration from Compute: For CPUs, the OS has the flexibility to assign processes to specific cores. If the load is low, the OS can assign all the processes to a few CPU cores and shut off the others to save energy. This is known as "core parking" and was leveraged *e.g.*, in [22, 25, 29].

Network: While shutting off entire pipelines could save more power than rate adaptation, it is not trivial to achieve due to the fixed mapping between input ports and processing pipelines; *i.e.*, an incoming packet on a given port must be processed by the pipeline this port is attached to.

As in § 4.2, one solution is to add a layer of indirection with circuit switches, but, this time, *inside* the router: between the physical ports and the ASIC (Fig. 5). Instead of keeping all the ASIC pipelines active, one can use a circuit switch to redirect traffic to only some, enabling turning the others off.

Challenges: While promising power-wise and conceptually simple, there are several challenges with this approach.

What is the latency cost? Ports taking turns being connected to the pipeline induces some delay during which incoming packets must be buffered. This could be done internally by using electrical circuit switches with small buffers, which would also hide from the application the time required to reconfigure the circuit switch. Moreover, electrical circuit

switches feature faster reconfiguration times than optical ones, which is important here. Another option would be to use Ethernet pause frames to buffer the traffic at the sender.

Is the addition worth it? While this new layer of indirection would allow energy savings by turning off ASIC pipelines, it comes at the cost of additional hardware. We postulate that the power cost of a pure circuit switch, either electrical or optical, would be small because a circuit switch does not do any processing; it just redirects signals from one port to another. For example, a free-space OCS only consumes energy for controlling the mirrors that redirect the optical signals. However, if we add buffers to an electrical circuit switch, as suggested above, the power footprint would increase, as signals would need to be decoded, written in the buffer, and re-encoded for processing by the ASIC.

Which pipeline to turn off, and when? Assuming pipelines are equivalent, one only needs to decide how many should be active. This can be done in a reactive manner: turn off a pipeline when the total throughput becomes small enough to be supported by the remaining ones, and vice versa. The challenge here is to be able to turn a pipeline on quickly enough to react to an increase in demand without inducing packet losses. Conversely, one can leverage the predictability of ML training workloads to orchestrate when pipelines are turned on and off based on when traffic is expected.

4.5 Going further: Redesigning the ASIC

The previous proposals aim to fix power proportionality in existing designs with better software, hardware configurations, or additional components. One could go further and rethink the entire design from scratch with power optimization as a primary objective. How would things change?

Packet processing mainly reads from memory, but writes little. This provides an interesting potential for distributing load across multiple processing units with limited overhead. A design with more but smaller units makes it easier to turn some of them off to match the current load, thus improving power proportionality. Fine-grained power gating could be enabled by more distributed network processing unit designs based on many small pipelines, chiplets, or similar.

There is another promising trend to consider: the arrival on the market of co-packaged optics switches [7] and silicon photonics [32]. These will allow bringing the optical to electrical conversion very close to the ASIC—inside the router—instead of taking place in the transceivers. For starters, this would make it trivial to include an OCS component in the design, as suggested in § 4.4. Moreover, this opens potential for performing some of the router's functionality directly in the optical domain; an idea already prototyped for processing inference in SmartNICs [36]. *E.g.*, one could envision performing the IP look up in the optical domain, which could save the optical to electrical to optical conversion altogether!

Designing an ASIC is a major enterprise, but the potential savings are worth considering for a sustainable digital future.

5 Conclusion

In this paper, we discussed the benefits of better power proportionality in networking hardware, even in the context of ML training, where compute power dominates (§ 3). We believe that the potential power savings or performance speedups should trigger some renewed interest in addressing networking power proportionality, both from the research community and hardware vendors. To initiate and foster this line of work, we took inspiration from methods applied to computing hardware today to sketch different directions to explore to improve the power proportionality of networking hardware (§ 4), from incremental improvements to network operating systems to entire ASIC redesign.

References

[1] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, and Hugh Williams. 2020. Sirius: A Flat Datacenter Network with Nanosecond Optical Switching. 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*. ACM, Virtual Event USA, 782–797. doi:10.1145/3387514.3406221

[2] Wenlei Bao, Changwan Hong, Sudheer Chunduri, Sriram Krishnamoorthy, Louis-Noël Pouchet, Fabrice Rastello, and P. Sadayappan. 2016. Static and Dynamic Frequency Scaling on Multicore CPUs. *ACM Transactions on Architecture and Code Optimization* 13, 4 (Dec. 2016), 1–26. doi:10.1145/3011017

[3] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. 2019. Energy and Power Efficiency. In *The Datacenter as a Computer: Designing Warehouse-Scale Machines*, Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan (Eds.). Springer International Publishing, Cham, 99–127. doi:10.1007/978-3-031-01761-2_5

[4] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. 2019. *Energy and Power Efficiency*. Springer International Publishing, Cham, 99–127. doi:10.1007/978-3-031-01761-2_5

[5] Ricardo Bianchini, Christian Belady, and Anand Sivasubramaniam. 2024. Data Center Power and Energy Management: Past, Present, and Future. *IEEE Micro* 44, 5 (Sept. 2024), 30–36. doi:10.1109/MM.2024.3426478

[6] Damien Borgetto, Henri Casanova, Georges Da Costa, and Jean-Marc Pierson. 2012. Energy-Efficient Job Placement on Clusters, Grids, and Clouds. In *Energy-Efficient Distributed Computing Systems*. John Wiley & Sons, Ltd, Chapter 6, 163–187. doi:10.1002/9781118342015.ch6

[7] Broadcom. 2024. Broadcom Delivers Industry's First 51.2-Tbps Co-Packaged Optics Ethernet Switch Platform for Scalable AI Systems | Broadcom Inc. https://investors.broadcom.com/news-releases/news-release-details/broadcom-delivers-industrys-first-512-tbps-co-packaged-optics

[8] Ken Christensen, Pedro Reviriego, Bruce Nordman, Michael Bennett, Mehrgan Mostowfi, and Juan Antonio Maestro. 2010. IEEE 802.3az: The Road to Energy Efficient Ethernet. *IEEE Communications Magazine* 48, 11 (Nov. 2010), 50–56. doi:10.1109/MCOM.2010.5621967

[9] David de la Osa Mostazo, Pablo Armingol Robles, Óscar González de Dios, and Juan Pedro Fernández-Palacios Giménez. 2024. Lessons Learned from IP Routers Power Measurements and Characterization. In *2024 15th International Conference on Network of the Future (NoF)*. 245–253. doi:10.1109/NoF62948.2024.10741444

[10] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2025. DeepSeek-V3 Technical Report. arXiv:2412.19437 [cs] doi:10.48550/arXiv.2412.19437

[11] EIA. 2025. Electric Power Monthly - U.S. Energy Information Administration (EIA). https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=epmt_5_6_a

[12] Nicolas Fevrier. 2023. Saving Power on ACX7000 Series. https://community.juniper.net/blogs/nicolas-fevrier/2023/09/11/saving-power-on-acx7000-series

[13] FS.com. 2025. Ethernet Transceiver Modules and Cables. https://www.fs.com/c/ethernet-networking-3859

[14] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, Shuqiang Zhang, Mikel Jimenez Fernandez, Shashidhar Gandham, and Hongyi Zeng. 2024. RDMA over Ethernet for Distributed Training at Meta Scale. In *Proceedings of the ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 57–70. doi:10.1145/3651890.3672233

[15] Romain Jacob, Lukas Röllin, Jackie Lim, Jonathan Chung, Maurice Béhanzin, Weiran Wang, Andreas Hunziker, Theodor Moroianu, Seyedali Tabaeiaghdaei, Adrian Perrig, and Laurent Vanbever. 2025. Fantastic Joules and Where to Find Them. Modeling and Optimizing Router Energy Demand. In *25th ACM Internet Measurement Conference (IMC 2025)*. Association for Computing Machinery, Madison, Wisconsin, USA, 15. doi:10.1145/3730567.3732920

[16] Michael A. Laurenzano, Mitesh Meswani, Laura Carrington, Allan Snavely, Mustafa M. Tikir, and Stephen Poole. 2011. Reducing Energy Usage with Memory and Computation-Aware Dynamic Frequency Scaling. In *Euro-Par 2011 Parallel Processing*, Emmanuel Jeannot, Raymond Namyst, and Jean Roman (Eds.). Springer, Berlin, Heidelberg, 79–90. doi:10.1007/978-3-642-23400-2_9

[17] Olli Mämmelä, Mikko Majanen, Robert Basmadjian, Hermann De Meer, André Giesler, and Willi Homberg. 2012. Energy-Aware Job Scheduler for High-Performance Computing. *Computer Science - Research and Development* 27, 4 (Nov. 2012), 265–275. doi:10.1007/s00450-011-0189-6

[18] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. 2017. RotorNet: A Scalable, Low-complexity, Optical Datacenter Network. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 267–280. doi:10.1145/3098822.3098838

[19] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. 2008. Reducing Network Energy Consumption via Sleeping and Rate-Adaptation. In *5th USENIX Symposium on Networked Systems Design and Implementation (NSDI 08)*. https://www.usenix.org/conference/nsdi-08/reducing-network-energy-consumption-sleeping-and-rate-adaptation

[20] NVIDIA. 2025. NVIDIA Docs. https://docs.nvidia.com/networking/display/ConnectX7VPI/Specifications

[21] NVIDIA. 2025. NVIDIA H100 GPU Datasheet. https://resources.nvidia.com/en-us-hopper-architecture/nvidia-tensor-core-gpu-datasheet

[22] Nwe Zin Oo and Panyayot Chaikan. 2021. The Effect of Core Parking for Energy-efficient Matrix-Matrix Multiplication by Using AVX and OpenMP. In *2021 21st International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 307–310. doi:10.23919/ICCAS52745.2021.9649926

[23] Daniel Otten, Sebastian Neuner, and Nils Aschenbruck. 2023. On Modelling the Power Consumption of a Backbone Network. In *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, Rome, Italy, 1842–1847. doi:10.1109/ICCWorkshops57953.2023.10283615

[24] Daniel Otten, Sebastian Neuner, and Nils Aschenbruck. 2023. On Modelling the Power Consumption of a Backbone Network. In *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, Rome, Italy, 1842–1847. doi:10.1109/ICCWorkshops57953.2023.10283615

[25] Amy Ousterhout, Joshua Fried, Jonathan Behrens, Adam Belay, and Hari Balakrishnan. 2019. Shenango: Achieving High CPU Efficiency for Latency-sensitive Datacenter Workloads. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI '19)*. USENIX Association, Boston MA USA, 361–377.

[26] packetpushers.net. 2025. Demystifying DCN Topologies: Clos/Fat Trees - Part2. https://packetpushers.net/blog/demystifying-dcn-topologies-clos-fat-trees-part2/

[27] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, Chao Wang, Peng Wang, Pengcheng Zhang, Xianlong Zeng, Eddie Ruan, Zhiping Yao, Ennan Zhai, and Dennis Cai. 2024. Alibaba HPN: A Data Center Network for Large Language Model Training. In *Proceedings of the ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24)*. Association for Computing Machinery, New York, NY, USA, 691–706. doi:10.1145/3651890.3672265

[28] Sudarsanan Rajasekaran, Manya Ghobadi, and Aditya Akella. 2023. CASSINI: Network-Aware Job Scheduling in Machine Learning Clusters. arXiv:2308.00852 [cs] doi:10.48550/arXiv.2308.00852

[29] Ahmad Samih, Ren Wang, Anil Krishna, Christian Maciocco, Charlie Tai, and Yan Solihin. 2013. Energy-Efficient Interconnect via Router Parking. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, Shenzhen, China, 508–519. doi:10.1109/HPCA.2013.6522345

[30] Shailesh S.Deore, A. N. Patil, and Ruchira Bhargava. 2013. Energy-Efficient Job Scheduling and Allocation Scheme for Virtual Machines in Private Clouds. *International Journal of Applied Information Systems* 5, 1 (Jan. 2013), 56–60. doi:10.5120/ijais12-450842

[31] Yanling Shao, Chunlin Li, Jinguang Gu, Jing Zhang, and Youlong Luo. 2018. Efficient Jobs Scheduling Approach for Big Data Applications. *Computers & Industrial Engineering* 117 (March 2018), 249–261. doi:10.1016/j.cie.2018.02.006

[32] Sudip Shekhar, Wim Bogaerts, Lukas Chrostowski, John E. Bowers, Michael Hochberg, Richard Soref, and Bhavin J. Shastri. 2024. Roadmapping the next Generation of Silicon Photonics. *Nature Communications* 15, 1 (Jan. 2024), 751. doi:10.1038/s41467-024-44750-0

[33] Arun Vishwanath, Kerry Hinton, Robert W. A. Ayre, and Rodney S. Tucker. 2014. Modeling Energy Consumption in High-Capacity Routers and Switches. *IEEE Journal on Selected Areas in Communications* 32, 8 (Aug. 2014), 1524–1532. doi:10.1109/JSAC.2014.2335312

[34] Sharada Yeluri. 2024. Flexible Packet Processing Pipelines. https://community.juniper.net/blogs/sharada-yeluri/2024/03/28/flexible-packet-processing-pipelines

[35] Qingxia Zhang, Zihao Meng, Xianwen Hong, Yuhao Zhan, Jia Liu, Jiabao Dong, Tian Bai, Junyu Niu, and M. Jamal Deen. 2021. A Survey on Data Center Cooling Systems: Technology, Power Consumption Modeling and Control Strategy Optimization. *Journal of Systems Architecture* 119 (Oct. 2021), 102253. doi:10.1016/j.sysarc.2021.102253

[36] Zhizhen Zhong, Mingran Yang, Jay Lang, Christian Williams, Liam Kronman, Alexander Sludds, Homa Esfahanizadeh, Dirk Englund, and Manya Ghobadi. 2023. Lightning: A Reconfigurable Photonic-Electronic SmartNIC for Fast and Energy-Efficient Inference. In *Proceedings of the ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 452–472. doi:10.1145/3603269.3604821