UNIVERSITY OF CALIFORNIA SAN DIEGO

Towards Leaner Data Centers: Energy Efficiency and Carbon Savings through Network Optimization

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Yibo Guo

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGEMENTS

I would like to start by thanking my advisor, Professor George Porter, for his continuous support and guidance throughout my Ph.D. journey. His feedback and advice have been invaluable in shaping my research and my career. I am grateful for his mentorship and for the opportunities that he has provided me to grow as a researcher.

I would also like to extend my gratitude to the rest of my committee members, Professor George Papen, Professor Aaron Schulman, and Professor Alex C Snoeren, for their unique insights and feedback on my thesis work.

Ph.D. program is a long journey and thus I am grateful for having the supportive and welcoming SysNet group at UC San Diego. Thank you everyone for your support and companionship throughout the years. I would also like to thank our SysNet admin, Cindy Moore, for providing all the technical support for our research needs.

Last but not least, I would like to thank my family for their unwavering support and encouragement.

Chapter 2 is largely a reprint of the material as it appears in the Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '22), Yibo Guo, William M. Mellette, Alex C. Snoeren, and George Porter, Association for Computing Machinery, 2022. The dissertation author was the primary investigator and author of this paper.

Chapter 3 contains material from two different sources co-authored by the dissertation author. Each source is used in a different part of the chapter.

Section 3.1 contains material that appeared in the 1st Workshop on NetZero Carbon Computing (NetZero '23), Yibo Guo and George Porter, 2023. The dissertation author was an investigator and co-author of this material.

Section 3.2 contains material that appeared in the Poster session of 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI '23), Yibo Guo and George Porter, USENIX Association, 2023. The dissertation author was an investigator and co-author of this material.

Chapter 4 in part is currently being prepared for submission for publication of the material. Yibo Guo, Amanda Tomlinson, Runlong Su, and George Porter, 2024. The dissertation author was the primary investigator and author of this paper.

VITA

2013–2014	Course Assistant, University of Illinois at Urbana-Champaign
2014	Software Engineer Intern, Microsoft
2015	B.S. in Computer Science, University of Illinois at Urbana-Champaign
2015–2017	Software Engineer, Microsoft
2017–2018	Teaching Assistant, University of California San Diego
2019	M.S. in Computer Science, University of California San Diego
2020	Software Engineering Intern, Google
2022	Teaching Assistant, University of California San Diego
2018–2024	Graduate Student Researcher, University of California San Diego
2024	Ph.D. in Computer Science, University of California San Diego

PUBLICATIONS

William M. Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C. Snoeren, and George Porter. “Expanding across time to deliver bandwidth efficiency and low latency”. In: *17th USENIX Symposium on Networked Systems Design and Implementation*. NSDI ’20. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 1–18

Yibo Guo, William M. Mellette, Alex C. Snoeren, and George Porter. “Scaling beyond packet switch limits with multiple dataplanes”. In: *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies*. CoNEXT ’22. Roma, Italy: Association for Computing Machinery, 2022, pp. 214–231

Yibo Guo and George Porter. “A metric for factoring data movement into chasing the sun”. 1st Workshop on NetZero Carbon Computing. Montreal, Canada, Feb. 2023

Yibo Guo and George Porter. “Carbon-aware inter-datacenter workload scheduling and placement”. Poster session of 20th USENIX Symposium on Networked Systems Design and Implementation. Boston, MA, Apr. 2023

Yibo Guo, Amanda Tomlinson, Runlong Su, and George Porter. “The effect of the network in cutting carbon for geo-shifted workloads”. In submission. 2024

ABSTRACT OF THE DISSERTATION

Towards Leaner Data Centers: Energy Efficiency and Carbon Savings through Network Optimization

by

Yibo Guo

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor George Porter, Chair

The rapid growth of computing in data centers worldwide has led to a significant increase in energy consumption and carbon emissions. As the demand for cloud services continues to surge, data centers have become the backbone of the digital economy, supporting a wide range of applications from social media and e-commerce to scientific research and artificial intelligence. This exponential growth in data centers has driven the need for more servers, storage devices, and networking equipment, all of which consume substantial amounts of electricity. This in turn contributes to higher carbon emissions and environmental impact. To make things worse, the end of Moore's law and Dennard scaling has exacerbated scaling challenges in both computing units like CPUs and network infrastructure, creating a bottleneck for data center growth and efficiency.

In this thesis, I propose two new ways to address scalability challenges in data centers and to support future data center growth:

First, I propose P-Net, a novel datacenter network architecture that improves the efficiency and scalability of data center networks (DCNs). P-Net, or parallel dataplane network, employs multiple network planes to achieve higher bandwidth and lower latency, instead of relying on the free scaling of underlying chips in traditional networks. By explicitly leveraging network chip parallelism, P-Net enables linear scaling of network resources with respect to bandwidth, reducing energy consumption, monetary costs, and scalability challenges in DCNs.

I then focus on the growing energy and carbon footprint of computing units like CPUs and accelerators, and explore new ways to tackle these challenges by exploiting low-carbon renewable energy sources. In particular, I explore the feasibility of space-shifting computational workloads across data centers as a carbon-aware computing solution. By utilizing energy efficient fiber optics wide-area network (WAN) links, space-shifting can make better use of localized renewable energy sources that cannot be used otherwise, achieving the goal of lowering power demand on the grid and reducing carbon footprint. Through detailed analysis, I show that by selecting the optimal data center(s) to run workloads based on both local carbon intensity and WAN cost, space-shifting can be more effective than alternative time-shifting solutions.

Chapter 1

Introduction

The adoption of cloud computing has surged in recent years due to its flexibility, scalability, and cost-effectiveness at scale. Organizations of various sizes leverage cloud services for a variety of applications, from data storage and processing to hosting complex software solutions, including modern AI and ML applications like the popular ChatGPT, cloud-based AI services as well as most recently Apple offloading their end-device ML computation to the cloud [14]. This widespread use of cloud computing has led to a rapid increase in data centers that power these services.

Data centers form the backbone of cloud computing, providing the necessary computing and network resources to support cloud services. Large-scale data centers, which are typical across hyperscalers like AWS, Microsoft Azure and Google cloud, can house rows upon rows of server racks, storage systems, as well as networking equipment that enable seamless access to cloud-based applications and data.

Operating at such scale, data centers are estimated to be $10\times$ to $50\times$ more energy intensive than a typical commercial building [25], and can consume tens to hundreds of megawatts of power each. As of 2022, worldwide data centers are estimated to consume about 460 TWh, or 2% of global electricity usage, and are responsible for about $2-4\%$ of global carbon emissions [100, 53]. Furthermore, the energy consumption of data centers is expected to grow fast in the near future, driven by the increasing demand for cloud services and the rise of more energy intensive AI and machine learning applications [26].

This together with the urgent climate goals published by many countries, organizations and more importantly data center operators [42, 19], has led to increasing concerns about not only energy, but also environmental impact of data centers and the need to build more sustainable data centers.

In this dissertation, I explore solutions of future data center growth and sustainability from both energy efficiency and carbon emissions perspective.

1.1 Energy efficiency in data centers and data center networks (DCNs)

Traditionally, energy improvements in data centers have focused on improving the energy efficiency of computing resources, such as CPUs, DRAM, and GPUs. As shown by Barroso, Hölzle, and Ranganathan [17], in 2017, close to 80% of the energy consumed in a data center is used by computing elements like CPU and DRAM; approximately 10% is used by non-IT equipment like power and cooling infrastructure, followed by 5% used by network equipment, and the remaining 5% is split among storage systems and miscellaneous components. Therefore, improving the energy efficiency of computing resources can have a significant impact on the overall energy consumption of a data center.

In this dissertation, I focus on an often overlooked part of data center energy consumption: the network infrastructure. Network is a critical component of data centers that is always running and exhibits poor power proportionality. And although historically it has consumed a small fraction of the total energy, this percentage is increasing from 5% in 2017 to double digits, because of the increased demand from data center applications as well as slow improvements in network switch power efficiency.

As computing in data centers continues to grow, so does the amount of data that's stored and accessed. Since storage in data centers today is disaggregated, more data to process means more network capacity to move data around. As shown in the growth of Google's production data center networks [92], the demand for network bandwidth is increasing rapidly and outpacing the advancement in bandwidth that commercial Ethernet switches provide. Moreover, modern applications like machine learning and big data analytics are increasingly bandwidth-intensive [52, 59], requiring more network capacity to support their operations. This means that data center operators are forced to deploy more network switches to meet the growing demand, which in turn consumes more energy.

To make things worse, network switch chips are facing similar scaling challenges like the end of Moore's Law and Dennard scaling in the CPU world. As shown in 2024 Ethernet roadmap [10], although switch speed continues to double approximately every other year, the power consumption of switches is also increasing rapidly. More concretely, Ethernet switch speed has grown from 640 Gbps in 2010 to 51.2 Tbps in 2022, resulting in a tremendous 80$\times$ improvement in bandwidth; but at the same time, the power consumption has also grown by 22$\times$. This means we have only achieved a 3.6$\times$ improvement in energy efficiency over the last decade. This poor power-per-watt scaling is especially true in the last few years, where the power consumption of switches increases by about 50% per generation for a 2$\times$ increase in bandwidth.

This together with the increasing demand in bandwidth, has led to a continuous increase in the proportion of energy consumed by network infrastructure in data centers [10].

To address the scalability issue in data center networks, researchers and operators have looked into solutions in several directions: 1) building domain-specific networks optimized for certain applications, 2) using energy-efficient optical network core and circuit-switched networks, and 3) better engineering of traditional packet switched networks.

1.1.1 Improving DCN energy efficiency via domain-specific networks

Traditional data center networks are built to support a diverse set of applications, from low-latency web services to high-throughput applications like big data analytics. These networks are often over-provisioned to support the worst-case scenario, which can lead to inefficiency in network resource utilization and energy consumption.

However, in cases where we know the characteristics of the applications, we can build domain-specific networks that are optimized for those applications. This has been historically done in the HPC community, where networks are optimized for scientific computing applications that have known traffic patterns and requirements, but more recently, in various machine learning clusters [110, 59].

By optimizing for a specific set of traffic patterns, this type of network can achieve better energy efficiency and resource utilization than traditional data center networks. However, the downside is that these networks are not as flexible as traditional networks, and may not be generalized to support arbitrary tenant applications like traditional data center networks.

1.1.2 Improving DCN energy efficiency via optical networks and circuit switching

In the context of generic data center networks, an alternative solution is to use optical network core and circuit switched networks. Optical networks transmit data using light signals over fiber optic cables, which are more energy efficient than sending electrical signals over copper cables. Optical networks have been shown to outperform electrical packet switched networks at higher speeds, and have been proposed as a promising solution to address the growing energy demands in data center networks in various academic and industrial settings [72, 73, 23, 16, 34, 109, 33, 40, 64, 81]. In fact, the Ethernet standard body is now calling for a closer integration of optics and switch dies to achieve optimal power efficiency [10].

However, this advantage also comes with a drawback: optical networks are not as flexible as electrical networks when it comes to routing, as packets cannot be parsed and routed in optical format. Instead, circuit switching is used to establish a path between the source and destination before transmitting data, which can introduce latency and routing challenges. To address this, researchers have proposed various solutions that generally uses a hybrid electrical-optical network architecture to handle such traffic [34, 109, 33, 40, 64, 73], although alternative fully optical network designs have also been proposed [72, 23, 16, 22].

Even though both sets of solutions have generally been shown to outperform traditional electrical packet switched networks in terms of energy efficiency and bandwidth scalability, there remain challenges in its adoption in practice. For instance, the industrial adoptions of this type of network is limited to specific settings, e.g. a subset of the DCN like the core layer [81].

1.1.3 Improving DCN energy efficiency via better engineering of existing packet switched networks

A third type of solution is to improve the energy efficiency in existing packet switched networks, which are the most widely deployed network architecture in data centers today. This incurs minimal changes to how we operate data center networks today, and can thus be more easily adopted by existing data center operators. Although modern high-speed switches are consuming more and more power, this is not necessarily the case if we optimize for not just performance, but also energy efficiency.

In this dissertation, I propose a solution that directly addresses the power scaling problem in existing Ethernet switches. In Chapter 2, I present P-Net [46], a novel data center network architecture that uses multiple parallel dataplanes to achieve linear scaling in energy consumption of high-speed switches. Such parallel network fabric has shown in practice, in LinkedIn's and Facebook's production networks, to enable $2-3\times$ reduction in chip count and power usage than the existing high-cost and high-power multi-tier chassis architecture.

Chassis architecture is the latest product in the evolution of modern Ethernet switches used in data center networks. Traditional data center networks are mostly based on the Fat Tree architecture [32]. As its name suggests, Fat Tree uses commodity switches to build a tree-like network topology in order to achieve bandwidth scalability. Because of the exponential increase in cost as the number of tiers grows, it is preferable to use the largest radix available in commercial switches. At the same time, the slow increase in the underlying switch chips' bandwidth and the increasing demand in per-port speed have led to the adoption of multilane

Ethernet switches, where multiple ports are grouped together to provide higher speed. Although this addresses the per-port speed demand, it also causes the switch radix to decrease if using a single chip per switch, which significantly hurts the scalability of Fat Tree-based networks. To avoid increasing the number of tiers in Fat Trees, the industry has been moving towards using chassis switches, where multiple layers of network chips are packed into a single chassis switch to achieve both high per-port speed and high radix.

This leads to the main drawback of this approach, which is that chassis switches use more than a linear number of network chips to support higher bandwidth, and so does the cost and power consumption. In P-Net, I show that instead of using multiple tiers and more than a linear number of switch chips to support higher per-port speed, we can opt for the lower per-port speed at native chip radix, and instead use multiple network planes to achieve the same total bandwidth. In fact, we see similar trends in the industry, where network operators like LinkedIn [119] and Facebook [12] opt for a parallel fabric approach to realize a $4\times$ increase from 100 G to 400 G, instead of using a single 400 G switch chip, for the same cost and power saving reasons. This is similar to the parallel Fat Tree architecture in P-Net, which first appeared in P-FatTree [74].

1.2 Carbon emissions of data centers

For the second part of this dissertation, I look beyond energy consumption and focus on the more urgent carbon emissions problem in data centers. This is because there is now a more pressing need to reduce the carbon emissions to meet climate-related goals in limited timeframe, e.g. by 2030 [42, 19]. Although any underlying improvement in energy efficiency will translate to proportional decrease in carbon emissions, the rate of such improvement is unlikely to be sufficient to meet these climate goals in time. Since data centers are becoming a major energy consumer and thus carbon emission producer, we need to explore more direct solutions to reduce the carbon emissions of today's data centers and to build more sustainable low-carbon data centers in the near future.

A key mechanism to reduce carbon emissions from data centers (and electricity consumers in general) is the adoption of low-carbon renewable energy sources. Renewable energy, such as solar and wind power, is becoming more cost-effective and widely available. For instance, in California, a leading state in renewable energy adoption, the total amount of solar and wind energy has increased rapidly over the past decade. But due to the intermittent nature of these energy sources, and the mismatch between energy supply and demand in terms of time and location, these low-carbon renewable sources may not always be fully utilized. Furthermore, limited battery storage and grid transmission capacity further restrict their potential use at later times or different locations. This results in an increasing curtailment of renewable energy, meaning that excess renewable energy cannot be used and is ultimately wasted [13].

1.2.1 Adopting renewable energy in data centers via time- and space-shifting

Data centers are in a unique position to help reduce the curtailment of renewable energy, as they can adjust their workload to match the availability of these renewable energy sources. This is the fundamental idea behind carbon-aware computing, which is further divided into two main approaches: time-shifting and space-shifting. As the names suggest, these two approaches directly address the renewable energy variation problem in time and space, respectively.

Time-shifting has been proposed as a promising approach to reduce the carbon footprint of data centers, as they are relatively simple and straightforward. As its name suggests, time-shifting reduces the carbon emissions of computing workloads by shifting the workload to times when the *carbon intensity*, or per-unit-electricity carbon emissions, of the grid is low. This has been demonstrated in various academic and industrial settings [83, 111, 4, 50]. However, the effectiveness of time-shifting can be limited by 1) carbon intensity variation or dynamic range at a single location, 2) workload flexibility or deadline limiting how long it can be delayed, 3) the amount of flexible workloads and spare capacity within a datacenter, and 4) access to low-carbon renewable energy, which are limited to a subset of today's data centers. Because of these limitations, we see that in practice, time-shifting has shown to have limited carbon savings in Google data centers [84].

Alternatively, space-shifting is another carbon-aware computing solution that involves moving the workload to data centers with lower carbon intensity. This approach theoretically can achieve much higher carbon savings than time-shifting, due to the large variation of renewable energy availability *across different data center locations* [6, 18, 82], as well as the lack of sufficient battery and transmission capacity to store and move electricity through the grid [88, 80, 18]. Prior work [122] has demonstrated the large potentials of space-shifting across data centers, specifically in reducing the renewable curtailment and greenhouse gas emissions. Zheng, Chien, and Suh show that by moving workloads from US east coast, which is a low-renewable and high-carbon region powered mainly by coal, to US west coast, where there is abundant low-carbon solar energy, data center operators can make use of $40-90\%$ of the excess renewable energy in California that cannot be used otherwise, and dramatically reduce the carbon emissions of these computing workloads.

In order to realize such carbon savings in space-shifting, we need a massive infrastructure to support workload migration across geographically distributed data centers. Fortunately, we have already built a massive infrastructure that connects all the data centers in the world, which is the wide-area network, or WAN for short. WAN has been historically used by data center operators to connect different data center locations, in order to support disaster recovery, backup, load balancing, etc. In addition, the physical fiber deployment has been provisioned with lots of spare capacity to allow for future growth (i.e. dark fibers), so the marginal capital cost of adding more capacity is relatively low. And lastly, because of its use of energy-efficient fiber-optics networks, the energy cost of data transmission over the WAN is much lower than transmission loss of renewable energy over the grid. Based on these factors, it is natural to leverage and extend this existing WAN infrastructure to support carbon-aware space-shifting, for the ultimate goal of reducing the carbon emissions of computing workloads in data centers.

In this dissertation, I explore the feasibility of such space-shifting approach as a carbon-aware computing solution, by first presenting the high-level architecture and challenges in Chapter 3, and then quantitatively modeling and evaluating the network carbon cost in detail in Chapter 4.

1.2.2 Operational vs embodied carbon emissions

Note that in addition to *operational* carbon emissions, or the carbon emissions from the electricity consumed by running the data center, there are also *embodied* carbon emissions, mainly the carbon emissions from the manufacturing and end-of-life process. The exact ratio of operational to embodied carbon emissions is still an open question, due to the lack of complete carbon data across the lifecycle of a device and the complexity of the supply chain. Furthermore, reducing embodied carbon emissions involves more business decisions and policy changes, requiring a fundamental shift in the way we design, manufacture, use and recycle data center equipment across the entire industry. These are all important topics, but are beyond the scope of this dissertation.

1.3 Dissertation overview

In this dissertation, I study two aspects of sustainable data centers: 1) improving the efficiency and scalability of data center networks (DCNs) and 2) reducing the carbon emissions of computing workloads through workload migration over the wide-area network (WAN).

For the first half of the dissertation, I focus on the scaling challenges in data center networks, and in particular, I look into how we can improve the energy efficiency and scalability of existing packet switched networks. In Chapter 2, I present P-Net, a new data center network architecture that directly addresses the non-linear scaling of existing chassis-based Ethernet switches. I show that instead of using multiple tiers and more than a linear number of switch chips to support higher per-port speed, we can opt for a lower per-port speed and use multiple network planes to achieve the same total bandwidth at per-host- and network-level. By explicitly leveraging network chip parallelism, P-Net enables linear scaling of network resources with respect to the number of hosts, reducing energy consumption, cutting monetary costs, and alleviating scalability challenges in DCNs.

For the second half of the dissertation, I move onto discussing the carbon emission aspects of data centers. In Chapter 3, I perform an initial study on carbon-aware space-shifting in data centers from a global perspective, based on the trend of increasing renewable energy deployment, and present at a high level the design of a geo-distributed carbon-aware workload scheduling and placement system. I then propose a detailed network carbon cost analysis over the WAN in Chapter 4, which is crucial in determining the benefits of carbon-aware space-shifting solutions, and evaluate the carbon savings of such solutions in a practical setting.

Finally, I conclude the dissertation with a summary of my contributions and potential future research directions in Chapter 5.

Chapter 2

Parallel networks: Scaling beyond packet switch limits with multiple dataplanes

In this chapter, I will dive into the scaling problem of existing Ethernet packet switches, specifically chassis switches used in fat trees. Chassis switch is the latest iteration of Ethernet switch that provides high speed and high radix, which helps achieve the bandwidth and scalability goals in data center networks, respectively. It does so, however, at the cost of higher power consumption and equipment cost.

Chassis switch comes into play when Ethernet switches no longer keep up with the increasing demands in *scale-out* networks. Traditional *scale-out* networks enable network operators to translate improved link and switch speeds directly into end-host throughput. Unfortunately, limits in the underlying CMOS packet switch chip manufacturing roadmap mean that NICs, links, and switches are not getting faster fast enough to meet demand. Chassis switches although can make up for this by using multiple chips per switch to increase the capacity, it does so at the cost of much higher chip count and power consumption. As a result, operators have introduced alternative, *parallel* fabric designs in the core of the network that deliver N-times the bandwidth by simply forwarding traffic over any of N parallel network fabrics, which has shown to achieve $2-3\times$ reduction in hop count, chip count, and power consumption.

In this work, we consider extending this parallel network idea all the way to the end host. Our initial impressions found that direct application of existing path selection and forwarding techniques resulted in poor performance. Instead, we show that appropriate path selection and forwarding protocols can not only improve the performance of existing, homogeneous parallel fabrics, but enable the development of heterogeneous parallel network fabrics that can deliver even higher bandwidth, lower latency, and improved resiliency than traditional designs constructed from the same constituent components.

Both homogeneous and heterogeneous parallel fabrics can achieve similar power and cost savings compared to the chassis-switch-based networks, showing promising solutions for future data center networks based on packet switches.

2.1 Introduction

Bandwidth-hungry applications demand ever more from datacenter network fabrics. Starting from "data-intensive" applications like MapReduce [27] and Spark [117], moving to graph-traversal systems [24], and on to emerging large-scale machine learning training [75], capacity requirements only continue to increase. While standards bodies and device vendors continue to deploy faster NICs, links, and switches, the pace of these deployments has slowed. As a result, there are periods of time where datacenter bandwidth needs exceed the capabilities of commodity network gear. To temporarily bridge this gap, datacenter operators like Facebook [12] and LinkedIn [119] have chosen to deploy *explicitly parallel* backplanes by connecting top-of-rack (ToR) switches with multiple disjoint replicas of their fabrics (in both cases four 100-Gb/s fabrics, with LinkedIn's version shown in Figure 2.1), delivering higher capacity without increasing the link rates of constituent components. This approach also allows operators to not only reduce their power consumption [119], but also lower the numbers of chips, boxes, and links required to build the network, as well as the number of hops between end hosts [12].

We assume that there will continue to be an "ebb and flow" pattern between the expected bandwidth required by the operators and the delivered bandwidth provided by commodity networking equipment. To meet bandwidth demands during the "ebb" phase of this cycle, we

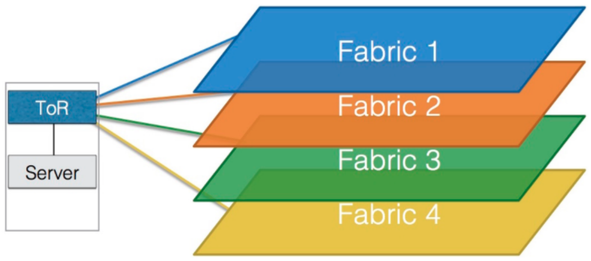


Figure 2.1. LinkedIn's 4-way parallel network [119]. Each fabric implements a 100-Gb/s fat tree [32] and is separate from one another except at the ToR switches.

propose a new class of *flattened* network topologies where each host—as opposed to ToR—is connected to *N* different disjoint network *planes*, each of which has its own set of switches and links connecting it to the other hosts in the network. We call these networks Parallel Dataplane Networks (P-Nets). Once a packet leaves the host in a P-Net and enters a given network plane, that packet cannot move to another network plane until it reaches its destination. This separation enables operators to linearly scale bandwidth by deploying multiple disjoint copies of their network—without having to insert switches or links between planes—lowering cost and energy demands.

However, the result is an explosion in the number of paths between end hosts, since not only does each network plane have multiple paths between a source and destination, but now there are multiple planes to choose from as well. Traditional fat tree networks already have multiple, equal-cost paths between hosts, and achieving full bisection bandwidth requires making effective use of them all [32]. Both operators and academics have developed a wide variety of techniques to approach the theoretical optimum in practice, with varying degrees of success. As we show in simulation, these techniques frequently struggle to achieve full performance when faced with the even larger set of choices in parallel network fabrics. Indeed, under certain conditions approaches like ECMP [3] barely leverage the added physical capacity.

We show that extracting the full, physical capacity of massively parallel P-Nets in practice requires explicitly striping traffic across the multiple planes in a strategic fashion. In particular, by using MPTCP [112] to multiplex flows across a bounded set of shortest paths, we are able to deliver performance similar to a traditional, single-plane scale-out network with upgraded link speeds. Not only can this additional sophistication unlock the extra capacity of additional, parallel network planes, but we observe that it also makes it possible to consider a further evolution in fabric topologies: parallel networks where the various forwarding planes are not simply identical copies of each other—as is inherent in a fat-tree-based approach—but instead, provide explicitly diverse physical connectivity. We refer to these networks as *heterogeneous* parallel networks. We further discover that such heterogeneity results in path-length variations across network planes, and by routing traffic over network planes with shorter paths to a given destination, we can reduce latency and bandwidth consumption, thereby also improving flow throughput. Even though such heterogeneous networks may be more complex to build, we find that recent optical interconnect technologies [89, 31] can dramatically simplify these operations and also lower power usage. Indeed, the gap between bandwidth demands and the delivered bandwidth of switch hardware can serve as an opportunity to explore alternative network architectures and protocols focused on improving application performance and efficiency.

In this chapter, we carry out a study of parallel homogeneous fat trees as well as heterogeneous expander-based architectures. We show the improved scalability in network equipment usage (chips, switch boxes and links) and power consumption that such parallel networks can provide, and also explore issues including routing, forwarding, and transport protocol performance across different degrees of parallelism. The primary contributions of this work are: (1) a study of P-Net, a class of networks that achieves high bandwidth by using multiple network planes consisting of lower-speed, but also lower-cost-and-power commodity switches and cables, (2) examples of suboptimal performance when adopting naive approaches to path selection and forwarding, and (3) a quantitative analysis of micro- and macro-application performance using both synthetic traffic and real datacenter flow traces, showing the benefits that heterogeneity can bring to parallel networks.

2.2 Motivation and background

The design of datacenter networks has been shaped by a combination of network bandwidth requirements from end hosts and the availability of high-speed merchant silicon packet switch chips. A driving factor in their design is ensuring that improvements in packet switching speed can be translated into increased end-host bandwidth. In this section, we describe this evolution and focus on recent scaling limits as motivation for deploying P-Nets.

2.2.1 The limitations of scale-out networks

Originally credited to Charles Clos [21], the observation that large switch fabrics can be composed of relatively small and inexpensive switches became relevant in datacenter network architecture with the advent of merchant silicon switch chips [32]. The structure of a folded-Clos network can be characterized by the number of tiers of switch chips that it requires, and how the chips are packaged into boxes. These design choices then dictate the number of hops a packet must traverse. Each additional tier incurs cost, power, latency, and cabling complexity, making it desirable to use the largest-radix commodity switches available.

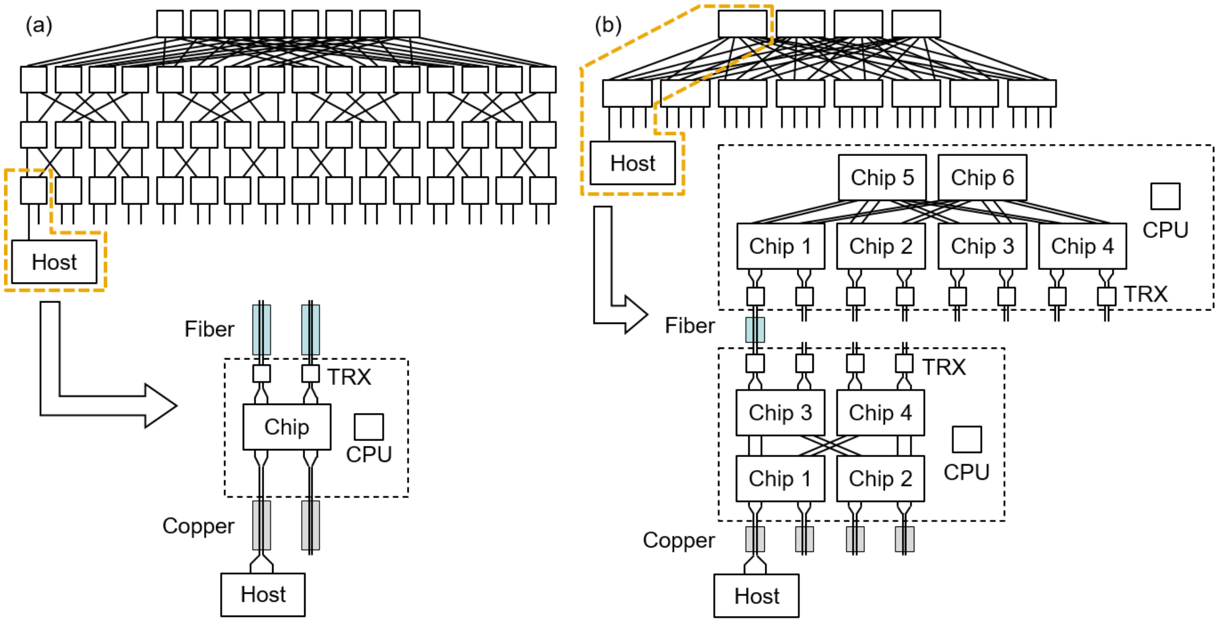


Figure 2.2. A (a) single-channel traditional and (b) single-channel chassis-based fat tree.

Figure 2.2(a) shows a small-scale illustration of a traditional folded-Clos or fat tree topology built from 4-port switches, with 32 end hosts and four tiers. As a more realistic—but difficult to illustrate—example, the components required to build an 8,192-end-host network out of 16-port switches are listed in the first row of Table 2.1. While small compared to today's largest networks, we use an 8,192-end-host exemplar network here because it allows for an "apples to apples" comparison between designs.

Table 2.1. Component counts for the two serial fat tree architectures shown in Figure 2.2 and parallel fat tree architecture shown in Figure 2.4. All networks have the same bisection bandwidth, with links in the 8× parallel networks optimized for deployment (Section 2.6.1).

Architecture	**Tiers**	**Hops**	**Chips**	**Boxes**	**Links**	**Transceivers**
Serial (scale-out)	4	7	3,584	3,584	24.6 k	49.2 k
Serial chassis	2	7	3,584	192	8.2 k	16.4 k
Parallel 8×	2	3	1,536	192	8.2 k	16.4 k

As shown in the first row of Table 2.1, traditional fat tree designs have several shortcomings, including the deployment and maintenance overhead of many (often long, fiber-optic) links and the replication of packaging and ancillary hardware including CPUs, fans, power supplies, etc.

2.2.2 The adoption of chassis switches

The disadvantages of traditional fat-tree designs led several industrial players to design and build chassis-based fat trees [103] in which multiple switch chips are integrated into a common box, known as a chassis, and connected using energy- and cost-efficient copper backplane traces. By increasing the radix and thus the density of switching capacity, this architecture requires fewer optical transceivers and long fiber runs which reduces total hardware, power, and deployment costs. Figure 2.2(b) illustrates a 32-host fat tree built with a chassis architecture, including how switch chips are connected inside aggregation and spine chassis. This chassis-based architecture has been a critical factor in enabling large-scale datacenter deployments, where the costs of a traditional fat tree would be infeasible [15, 92].

The second row of Table 2.1 shows the components required to build an 8,192-node network out of 128-port chassis. Spine chassis use 24 16-port chips in a 3-stage internal Clos. Aggregation chassis do not need to be non-blocking, and are built with 16 16-port chips in a 2-stage topology. Despite the blocking aggregation chassis, the network as a whole retains the non-blocking property of Clos topologies, a fact leveraged in production networks [92]. Only two tiers of switch chassis are required, reducing the cabling by 1/3. While the number of switch chips remains constant, the number of discrete switch boxes is reduced by an order of magnitude.

2.2.3 Scaling beyond current chip limits

While the chassis architecture affords cost and power savings over a traditional scale-out fat tree, it has its own limitations. First, each chassis comes with a high power density, which presents scaling challenges as link speeds increase. (E.g., Facebook had to re-design their switches to fit in a hard 1750 W per-rack power limit [96].) Further, because the chassis architecture is simply a re-packaging of the traditional fat tree, it is ultimately subject to the same underlying scaling limitations in terms of switch chips and hops.

To circumvent these limitations, industrial datacenter network operators have recently begun to introduce parallelism to the core of their network designs, exposing the underlying per-port bandwidth to ToR switches rather than using chassis switches to implement a higher-bandwidth single link abstraction [12, 119]. In existing designs, hosts are connected to ToRs, which are then connected to multiple, independent network planes. StarDust [124] proposes a slightly different design that also opted for lower per-port bandwidth and higher radix, but instead form a single network in the core and rely on cell-based forwarding and simpler switch hardware design to scale the network. In this chapter, we take this approach a step further, and consider what happens when end hosts are connected to each of the planes directly—i.e., ToRs are a member of only one plane.

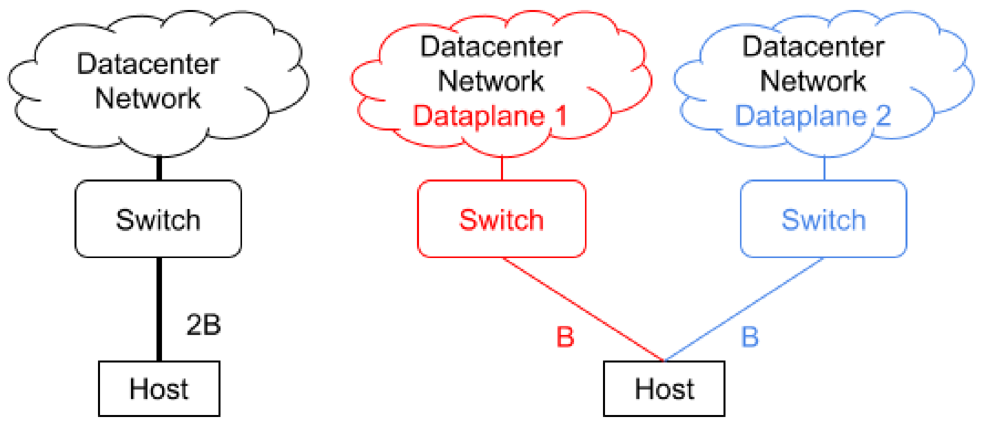

Datacenter Network
Switch
2B
Host
Datacenter Network Dataplane 1
Switch
B
Datacenter Network Dataplane 2
Switch
B
Host

Figure 2.3. A “serial” network (left) uses high-bandwidth links throughout the topology, but requires high-cost and high-power chassis switches. A P-Net (right) uses lower-bandwidth links, allowing each switch to be implemented with a single lower-cost and lower-power switch chip.

2.3 Parallel Dataplane Networks

The basic concept of Parallel Dataplane Networks (or P-Nets) is relatively straightforward: rather than building a single “serial” high-link-speed network, the network is architected as multiple, disjoint lower-link-speed forwarding planes, or *dataplanes* for short, as illustrated logically in Figure 2.3. P-Nets extend parallelism all the way to end hosts, where each host has connections to N dataplanes via N ToR switches, albeit at $1/N\times$ the bandwidth per connection. The important distinction between a serial network and a parallel network is this: each dataplane runs like a traditional serial network, but packets cannot cross dataplanes, as they are logically separate. This leaves the end host to decide which dataplane(s) to send its traffic through, and once a packet leaves an end host and enters a particular dataplane, it stays within the dataplane until reaching the destination host.

Even though each dataplane only provides a fraction of the total uplink bandwidth, end hosts can still make use of the full bandwidth by employing multipath solutions like MPTCP [112]. In this section, we present the design of P-Nets, show the difference between traditional serial networks and our parallel networks, and explain how to adapt end hosts and applications to make full use of these parallel dataplanes.

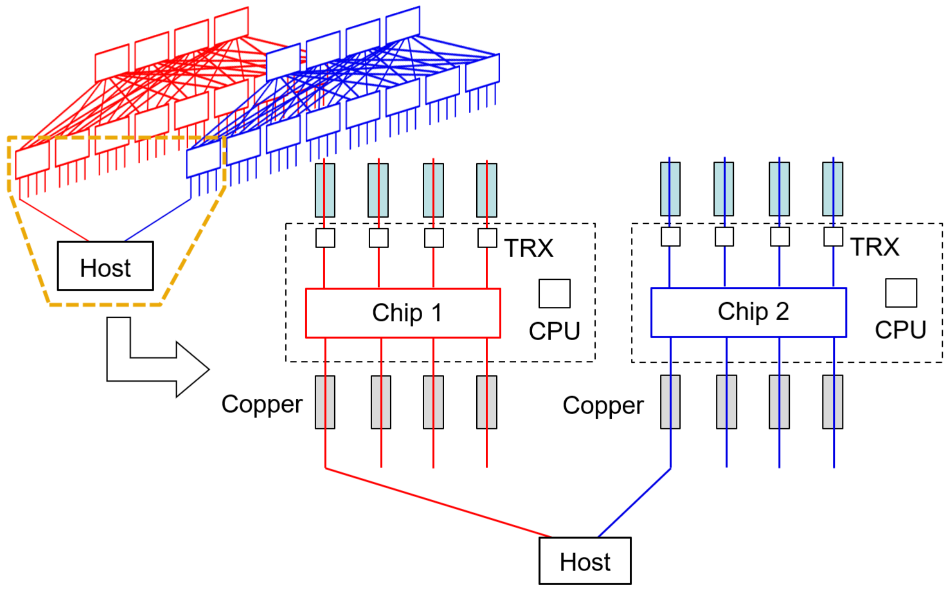


Figure 2.4. A two-way parallel fat tree/ homogeneous P-Net.

2.3.1 Homogeneous P-Nets/ Parallel fat trees

Perhaps the simplest class of P-Nets are those that retain the same connectivity and physical structure among the dataplanes. We refer to these as homogeneous P-Nets, and the most straightforward example is a parallel fat tree. Figure 2.4 shows a 32-host parallel fat tree with two dataplanes; each host connects to both the red and blue dataplanes as shown. This fabric is constructed using the same switch chips as the networks in Figure 2.2, but in a different configuration. Note that in scale-out and chassis designs, although each switch chip has eight inputs, or internal ports, they only act as four ports, because every two internal ports are grouped into one high-speed port. This comes at the cost of lower radix, and chassis switches compensate for this by using two tiers and *four* chips per switch to maintain the radix of eight. The two-way parallel fat tree shown in Figure 2.4, on the other hand, breaks these grouped ports out to support the same number of hosts, and compensates for per-host bandwidth by connecting both planes to each host.

The design of a parallel fat tree results in a fabric with the same switching capacity as multistage chassis networks, but with fewer switch chips and lower power density. The parallel topology also reduces the number of hops as it maintains the switch chips' native radix by not grouping ports together and does not require multiple tiers of chips as in chassis switches. The final row of Table 2.1 compares the component usage of an 8× parallel fat tree to traditional serial and chassis designs, all with an equivalent number of end hosts and bisection bandwidth. Comparing the parallel fat tree to the chassis-based design, we see that P-Net enables significant hardware and power savings, while also reducing the number of hops. Our analysis shows the same $2-3\times$ reduction in hop count, chip count, and power consumption as in [12]. Note that a naive implementation of a parallel fat tree can increase the number of physical cables, but by leveraging the homogeneity of the network, we can group multiple low-bandwidth links using cable bundles to significantly reduce complexity (details in Section 2.6).

2.3.2 Heterogeneous P-Nets

While parallel fat trees present an intuitive picture of parallel networks, P-Nets are not restricted to simply replicating identical copies of each dataplane. A prominent candidate for heterogeneous P-Nets is expander-graph-based networks. Due to their random [94] or pseudorandom [104] construction, we can create different instantiations for each dataplane, opening up new options for forwarding traffic.

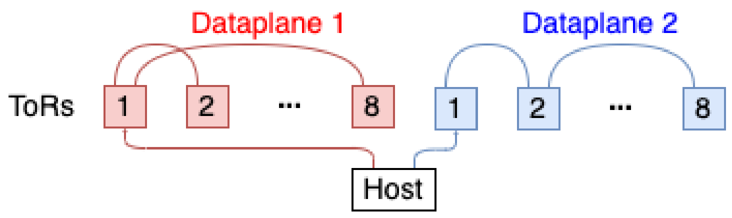


Figure 2.5. A two-way parallel heterogeneous expander.

Figure 2.5 shows a P-Net built with two expander-based dataplanes, where each dataplane instantiates a different expander network among the ToR switches. Serial expander graphs are known to have short path lengths [94], but applying P-Net to expanders yields additional benefits because the probability of having a short path between a particular pair of racks grows with each additional data plane realization. Hence, a heterogeneous P-Net has the potential for significant latency improvement over a homogeneous P-Net. We evaluate these benefits in Section 2.5 and address the practical issues of such heterogeneous topologies in Section 2.6.

2.3.3 Implications on switches

Today's network switches can be configured to provide either fewer ports at higher speed (e.g. 32 ports at 400 Gb/s per port) or more ports at lower speed (e.g. 128 ports at 100 Gb/s per port) [9, 76]. Serial network designs like the scale-out and chassis-based fat trees shown in Figure 2.2 configure switches for lower radix, requiring more tiers of switches leading to high cost, power, and hop count. P-Nets, on the other hand, configure the constituent switches with a higher radix, allowing a significant reduction in the number of switching tiers and commensurate savings in cost, power, and hop count. P-Nets' multiple planes require only a linear multiple of switch chips, whereas traditional and chassis-based fat trees need even more due to additional tiers and denser switches. This allows P-Nets to scale similarly to [124], as both approaches opt for higher radix of the underlying chip.

2.3.4 Implications on end hosts

Multiple uplinks to ToRs

While P-Nets can, in principle, have an arbitrary number of parallel dataplanes, in practice end hosts must have a sufficient number of uplinks to connect to all dataplanes. This can be achieved with multi-channel Ethernet links (details in Section 2.6.1), multi-port NICs [5, 62] or multiple NICs per server. For practicality, we limit the number of dataplanes to ≤ 8, similar to how today's high-speed (e.g. 400 G) ports aggregate up to 8 lower-speed ports. An even higher level of parallelism exacerbates the deployment complexity and often necessitates a generational improvement in the underlying silicon speed.

Utilizing multiple links in OSes

At the OS level, we expose multiple dataplanes to end hosts at the IP layer following reasons: 1) this is the default way of accessing multiple ports on Linux; 2) by using different IPs per dataplane, we can use existing end-host routing solutions like DARD and Fastpass [113, 79] on each dataplane; 3) applications can decide which dataplane(s) to send their traffic through by using the appropriate IP address(es); 4) we can reuse existing multipath transport like MPTCP [112] to seamlessly adapt deployed applications. End hosts can also quickly detect individual dataplane failures via link status and avoid using the broken dataplane(s), allowing graceful performance degradation.

End-host routing solutions provide OS direct access to routing information and can facilitate better flow placement decisions in P-Net. This approach also avoids the limited memory constraint on commodity switches in order to support routing over multiple dataplanes.

Application interaction

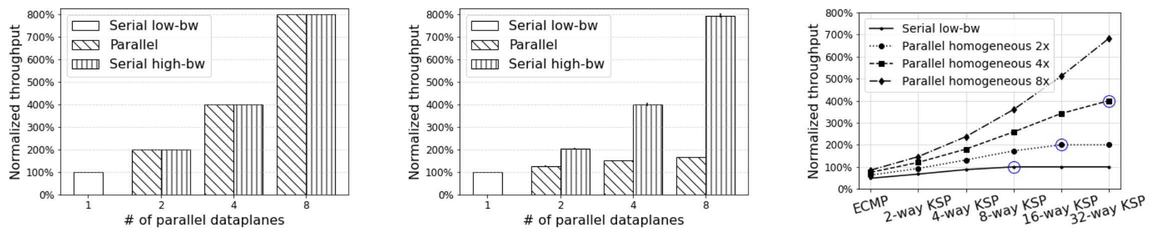
By default, round-robin is used for load balancing, but as we show in Section 2.5, applications with special needs like low latency and high throughput can choose to use low-hop-count dataplane and to send traffic across multiple dataplanes to achieve their respective goals.

In practice, end hosts are aware of the topologies of all dataplanes in P-Net, and thus can provide pseudo/proxy interfaces like “low-latency” single-shortest-path and “high-throughput” multipath interfaces. Applications/ flows can use special tags like traffic classes to choose how to take advantage of the multiple dataplanes in P-Net.

2.4 Forwarding traffic over multiple dataplanes

The performance of a network depends not just on its topology, but also on the way that traffic is forwarded over that topology. There have been numerous studies of path selection algorithms for different topologies, and in this section, we discuss how to adapt these approaches to P-Net.

To start with, we first considered adapting ECMP [3]. In this case, each end host selects, for each flow, one of the *N* parallel dataplanes using a hashing algorithm. We simulated all-to-all



(a) All-to-all throughput **(b)** Permutation throughput **(c)** Single-path vs multi-path

Figure 2.6. Fat tree ideal throughput with ECMP (a and b), and performance scaling using multipath (c). Circled points indicate the multipath level needed to saturate P-Net. Throughput normalized against serial low-bandwidth.

and permutation traffic using ECMP on parallel fat trees (we defer the details to Section 2.5.1), and Figure 2.6a and 2.6b show the achieved throughput plotted against that of a serial fat tree. We found that even though dense traffic patterns like all-to-all can fully saturate up to 8× parallel fat tree networks, sparser traffic patterns like permutation achieve minimal performance improvement when adding more dataplanes. Thus we concluded that naive ECMP-based routing is not adequate to fully exploit the combined capacity of these parallel dataplanes.

As single-path ECMP-based routing cannot fully utilize parallel dataplanes in P-Net, we next considered multipath routing and transport. Specifically, we looked at MPTCP [112] combined with K shortest paths (KSP) [43, 116] routing. The Jellyfish [94] work found that this provides a large improvement over ECMP + TCP in utilizing an increased number of paths. Figure 2.6c shows the throughput for the same permutation traffic in Figure 2.6b for parallel and serial fat trees, but this time using MPTCP and KSP, for various values of K. With MPTCP and KSP, sparse permutation traffic can now fully utilize the combined bandwidth of the parallel dataplanes. In fact, more parallel dataplanes demand higher multipath levels to saturate the network: 8-way multipath can fully utilize serial networks, but 2-dataplane P-Nets need 16-way multipath and 4-dataplane P-Nets need 32-way multipath.

Furthermore, we note that KSP is commonly employed in expander networks [94, 104], which are prime candidates of heterogeneous P-Nets, and Singla et al. have shown its good performance on serial expander networks [94]. Thus, we propose MPTCP + KSP as a promising approach for both homogeneous and heterogeneous P-Nets. We analyze its performance in heterogeneous P-Nets in Section 2.5, and propose a method for extracting the best performance for both short and long flows given MPTCP's limited ramp-up and convergence time.

2.5 Performance evaluation

Now that we have discussed how to build P-Nets, let's quantitatively evaluate how P-Nets perform relative to traditional single-dataplane networks. To evaluate the performance of P-Nets, we used a combination of synthetic workloads and real data center traffic distributions. The synthetic workloads consist of bulk and short flows, which represent the typical elephants and mice datacenter traffic, as well as application-like traffic patterns with multiple requests/responses. Real traffic traces include webserver/cache/Hadoop [87], datamining [44] and web search [7].

Depending on the traffic types, we use either linear programming (LP) solver [68] to measure the throughput, which allows us to scale much larger than packet simulators at high speed (100/400 G), or packet simulator *htsim* [51] to capture latencies and flow completion times. Except otherwise noted, we repeat each experiment at least five times (each time with a newly instantiated topology) and plot the standard deviations as error bars. We find little variation in most cases.

In our evaluations, we considered the following types of networks:

- **Serial low-bandwidth** network, which is a single network composed of (relatively) low capacity links. In our evaluation, we use 100 Gb/s links (i.e. 1×100 G).
- **Parallel homogeneous** network, or homogeneous P-Nets, which consists of *N* parallel dataplanes each with the same topology (e.g. fat tree). The links in each data plane run at 100 Gb/s.
- **Parallel heterogeneous** network, or heterogeneous P-Nets, in which each of the *N* data-planes implements a different topology (e.g. expander graph).

- **Serial high-bandwidth** network, which represents an ideal (but cost- and power-prohibitive network) with links running at $N \times 100$ Gb/s.

Our goal here is to compare traditional serial networks with their parallel (homogeneous/heterogeneous) versions; specifically, we want to see whether P-Net can achieve similar or close performance of an ideal serial high-bandwidth network, and what other benefits/drawbacks does adding parallelism to the network bring. We do not attempt to directly compare fat trees with expanders.

Note that although we choose 100 G networks as our baseline, most of the *throughput* evaluation results can apply to networks of arbitrary speed (25 G, 40 G, or more than 100 G) and their parallel or high-bandwidth equivalent, since the benefits of parallelism come from using multiple networks planes, i.e. the degree of parallelism, not the actual speed.

2.5.1 Microbenchmark

Bulk traffic throughput

One of the key questions we'd like to answer is whether parallel networks can achieve similar throughput as a serial high-bandwidth network. In this section, we compare the total throughput of flows under all-to-all and permutation traffic matrices.

We first measure the ideal throughput under no path constraint, which represents the total capacity of the network core. For this purpose, we run rack-level all-to-all traffic on a 128-rack fat tree and an equivalent Jellyfish (built with the same networking equipment, as described in [94]) using the LP solver. We plot the throughput on Jellyfish networks in Figure 2.7. Throughputs are normalized against serial low-bandwidth networks.

Because parallel heterogeneous networks using Jellyfish-like topologies have randomness [1] across the dataplanes, there may exist a shorter path for a given source/destination pair on another dataplane. Thus, each flow may consume less network capacity and thus, such

[1]This is the major difference and advantage of parallel heterogeneous expander networks over simply-larger-radix serial expander networks.

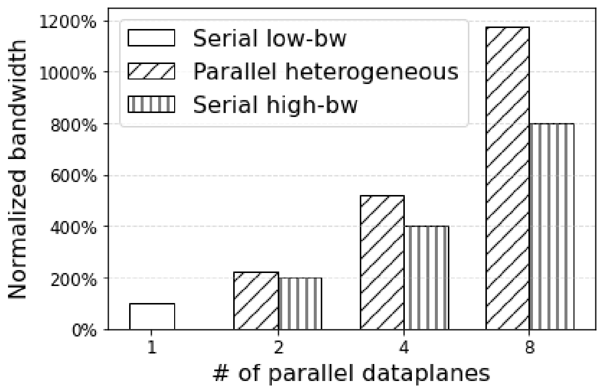
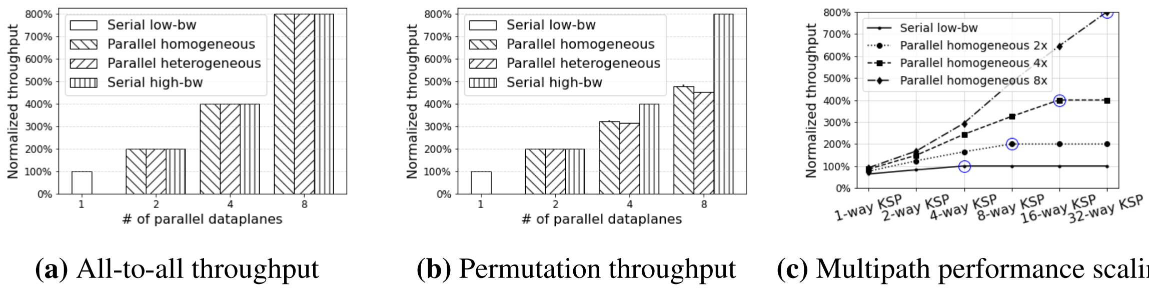


Figure 2.7. Ideal throughput on Jellyfish with *rack-level* all-to-all traffic.



(a) All-to-all throughput **(b)** Permutation throughput **(c)** Multipath performance scaling

Figure 2.8. Jellyfish ideal throughput with 8-way KSP (a and b), and performance scaling using multipath (c). Circled points indicate the multipath level needed to saturate P-Net. Throughput normalized against serial low-bandwidth.

heterogeneous parallel networks end up having even higher in-network capacity than their single-dataplane equivalent. Figure 2.7 shows that parallel Jellyfish networks can have up to 60% higher throughput than their serial high-bandwidth equivalent. Homogeneous P-Nets on both Jellyfish and fat tree topologies have the same throughput as serial high-bandwidth and are thus omitted.

Next, we take into account routing by simulating the ideal throughput with computed routes. This means we constraint the flows in LP solver to use the routes computed by ECMP or KSP. We ran all-to-all and permutation traffic on 1024-host fat trees and equivalent Jellyfish networks, and calculated the achieved throughput. The results are shown in Figure 2.6a and 2.6b for fat trees and Figure 2.8a and 2.8b for Jellyfish networks, respectively. We note that in both cases, dense traffic patterns like all-to-all can fully saturate all the parallel dataplanes, but for sparse traffic patterns like permutation traffic, standard ECMP for fat tree and default 8-way

KSP for Jellyfish (which has been shown to have good performance in serial expander networks) cannot fully utilize the increased bandwidth from multi-dataplane parallel networks. In parallel fat trees, ECMP improves the throughput minimally even with $8\times$ as many dataplanes; and in parallel Jellyfish, the default 8-way KSP can only achieve about 60% of the total bandwidth.

Consequently, we quantitatively evaluated the impact of multipath in P-Nets by varying the degree of multipath and measuring the throughput. We adjusted the routing parameter, ranging from single-path ECMP to K-way shortest paths for $K = 2, 4, 8...$ up to 32. The results are shown in Figure 2.6c for parallel fat tree networks and Figure 2.8c for parallel Jellyfish networks. As we can see, increasing the degree of multipath dramatically improves the achieved throughput, especially in the case of parallel networks. Furthermore, as highlighted in circles, more dataplanes demand a proportionally higher degree of multipath to fully utilize the combined bandwidth, i.e. P-Nets with N dataplanes need N times as many subflows. This is consistent with the intuition of using KSP in expander networks, which says more subflows are needed to saturate the increased number of uplinks/paths in expanders. In P-Net, the same reasoning applies: more dataplanes means more paths available; thus more subflows are needed to use the combined bandwidth of all dataplanes.

Short flow completion time

To capture the behavior of short flows, we set up a random permutation traffic pattern in a 686-host Jellyfish network using our packet simulator "htsim" [51], and measured the flow completion time as we vary the flow sizes from 100 kB to 1 GB. For this evaluation, we use P-Nets with four dataplanes.

Figure 2.9 compares the flow completion times (FCTs) of the four different types of networks. We also varied the degree of multipath and found that single-path routing generally gives the lowest FCTs in both serial networks whereas 4-way KSP gives the lowest FCTs in both parallel networks. Thus we plotted these results using these best settings for the respective networks.

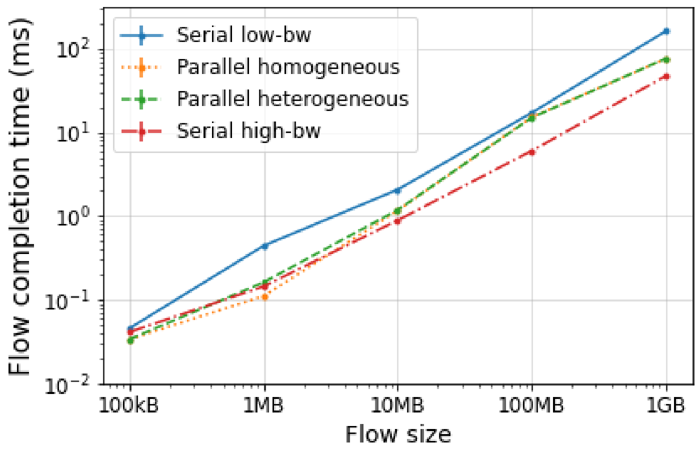


Figure 2.9. Small flow FCT with varying flow sizes. Parallel networks benefit very small flows since they can use more paths available in the parallel dataplanes without running into collision. Relatively large flow (100 MB+) starts to behave like steady state: flows on the smaller side like 100 MB cannot efficiently use multiple dataplanes due to slow MPTCP ramp-up time, whereas thus larger flows like 1 GB can take advantage of multiple dataplanes using MPTCP.

We note that for smaller flow sizes up to 10 MB, parallel networks have surprising advantages over serial networks and even outperforms serial high-bandwidth network. Upon deeper investigation, we found that these flows are small enough that they can finish before hitting full queues and causing retransmits, i.e. before reaching TCP steady state. In such cases, flows in parallel networks have the advantage of using more paths in TCP slow start. However, one should consider the various factors at play at such small scales, which include slow start behavior, the retransmit timeout (we tuned to 10 ms as suggested in DCTCP [7]), network load or switch queue utilization, etc. Also, as shown in [123, 29, 28], MPTCP can often hurt short flows, which happens here in serial networks. Thus, one should take care in using MPTCP for these really small flows (including sub-100 kB flows), and their exact behavior in datacenters could demand further tuning and studies.

As flows grow larger to 100 MB, they start to reach steady state and TCP/MPTCP was able to probe the bandwidth available across the dataplanes. Here, P-Nets have a smaller advantage over baseline serial low-bandwidth networks because MPTCP is slow to probe the optimal subflow bandwidth allocation at smaller time scales (only 100 MB on a 4×100 G network). Larger flows like 1 GB on the other hand can realize more speedup over serial low-bandwidth because MPTCP can probe bandwidth better.

To summarize, we note that in steady state: relatively small flows achieve less improvement in parallel networks, whereas larger flows start to behave like bulk flows and can benefit from using more paths in P-Nets. Thus we empirically choose 100 MB as a threshold for small versus large flows. Flows smaller than or equal to 100 MB benefit less from multipath using MPTCP and thus should use single-path routing; flows larger than or equal to 1 GB can significantly improve their performance by using multipath and thus should do so.

2.5.2 Simulated workloads

Small RPCs

As discussed in the previous section, MPTCP could hurt the performance of small flows. This is especially true for small RPCs that can be a few packets or even less than an MTU-sized packet. For these small packets, we choose to run single-path ECMP. Because of the random nature of expander graphs and potentially shorter paths in another dataplane for any given source and destination, there could be another way to improve RPC latency in such networks, i.e. by using shorter paths available in heterogeneous P-Nets.

To verify our hypothesis, we set up a packet simulation using a 686-host Jellyfish network and a ping-pong type RPC application. Again we use P-Nets with four dataplanes. Each host will send an RPC request of MTU size (1500 B) to a random destination server, wait for the response and measure the end-to-end request completion time over 1000 rounds.

Figure 2.10 shows the request completion time distribution in CDF and Table 2.2 summarizes the statistics.

The stepping curves come from the different hop count distributions in these networks. As we can see, serial networks and parallel homogeneous networks have identical hop count distribution as they use the same topology, whereas parallel heterogeneous networks use a mix of four different instantiations of Jellyfish. And as we suspected, for any given source and

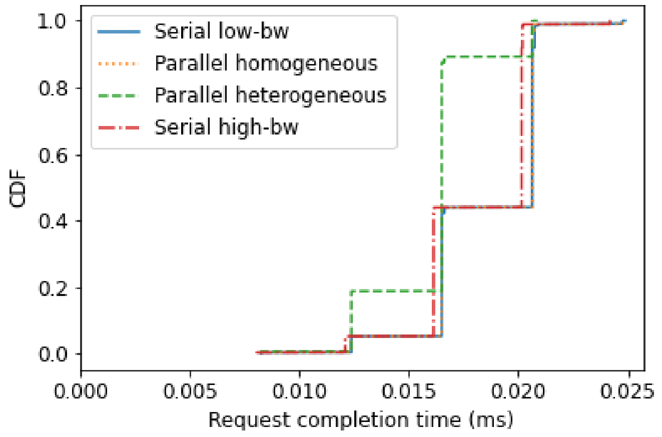


Figure 2.10. 1500 B RPC request completion time, single-path routing. Parallel homogeneous mostly overlaps with serial low-bw.

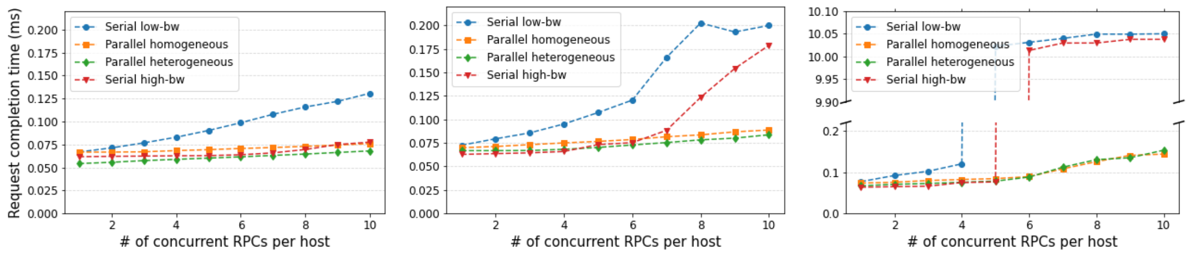
Table 2.2. 1500 B RPC request completion time statistics, using serial low-bw as baseline.

Network setup	**Median**	**Average**	**99%-tile**
Serial low-bw	100%	100%	100%
Parallel homogeneous	100%	99.2%	100%
Parallel heterogeneous	80.1%	86.6%	90.4%
Serial high-bw	98.1%	97.9%	97.4%

destination, there may exist a shorter path on the three additional dataplanes, compared with the other three types of networks. Thus parallel heterogeneous networks have the unique advantage in terms of lower hop count, which translates to lower latency and RPC completion time.

The minor difference between serial low-bandwidth and parallel homogeneous comes from the fact that parallel homogeneous networks have more paths, thus reducing the chance of flow collision given the same number of flows.

The serial high-bandwidth network has slightly less serialization delay per hop, as its links run at 400 G instead of 100 G in the other three networks. However, at 100 G, MTU-sized packets only take 1500 B/100 Gb/s = 120 ns; at 400 G, it's only $1/4$ of that. Thus serial high-bandwidth networks can only reduce the latency by 90 ns per hop. Such improvement in serialization delay is relatively small compared with the propagation delays in modern datacenters. Assuming 200 m per switch hop in the core, each hop will introduce a whole microsecond, which



(a) Median **(b)** 90%-tile **(c)** 99%-tile

Figure 2.11. Concurrent RPC request completion time. Note the broken axis in (c).

is 11× the serialization delay improvement in serial high-bandwidth networks. Thus overall, parallel heterogeneous networks can achieve much lower latency for small RPCs by using these shorter paths. As links become faster, the advantage of high-bandwidth networks diminishes, but propagation delay is fixed by the law of physics; thus the reduction in path length will further improve the end-to-end latency for small RPCs in high-speed networks.

In addition to single RPC evaluations, we also experimented with multiple concurrent RPCs. Since P-Net has more paths, we suspect that it is possible to handle more small RPC requests with less queuing behind other requests than in serial networks.

In this experiment, we use the same network setup, but with 100 kB-sized requests, and we vary the number of concurrent RPCs per host from 1 to 10. Figure 2.11 shows the median, 90 th and 99 th percentile of RPC request completion time. As we increase the number of concurrent RPCs, the request completion times also increase. Serial low-bandwidth suffers the most, as there are both limited bandwidth to drain these packets in the queue and a limited number of paths to avoid path collision that leads to queue buildup. Serial high-bandwidth networks reduce this problem only slightly by draining the queue faster. Parallel networks, on the other hand, have the advantage of 4× the number of paths, allowing concurrent RPC requests to spread across these separate links and queues. This results in both 1) less queue buildup and only a mild increase in request completion time, and 2) less chance to run into full queues, packet drops, and retransmits, as shown in Figure 2.11c.

Shuffle workloads

Inspired by Hadoop-type large-scale data analytics jobs, we simulated similar workloads involving loading data from remote racks, shuffling data among workers , and writing output data to replicas.

We simulated the Hadoop traffic of a sorting application in a 250-host cluster, in which we distribute 100 G data to 32 mappers and 32 reducers. Each mapper loads data in blocks of 128 MB, spreads the entries into 32 buckets, and sends each bucket to one reducer to merge and sort all entries in that bucket. We assume the entries are randomly distributed; thus, the shuffle stage consists of 32×32 flows of the same size, one per (mapper, reducer) combination. After a reducer completes sorting, it will write to a replica in a random rack. We configured our mappers and reducers to read/write 4 concurrent blocks at a time to avoid sending too many flows at once. For the read input and write output stages, each flow corresponds to one block, which is 128 MB. For the shuffle stage, each flow gets $1/(32 \times 32)$ of the total 100 GB of data, which is roughly 100 MB. Since these are relatively short flows in our $100/400$ G network setup as we showed earlier, we choose to run single-path routing for these flows.

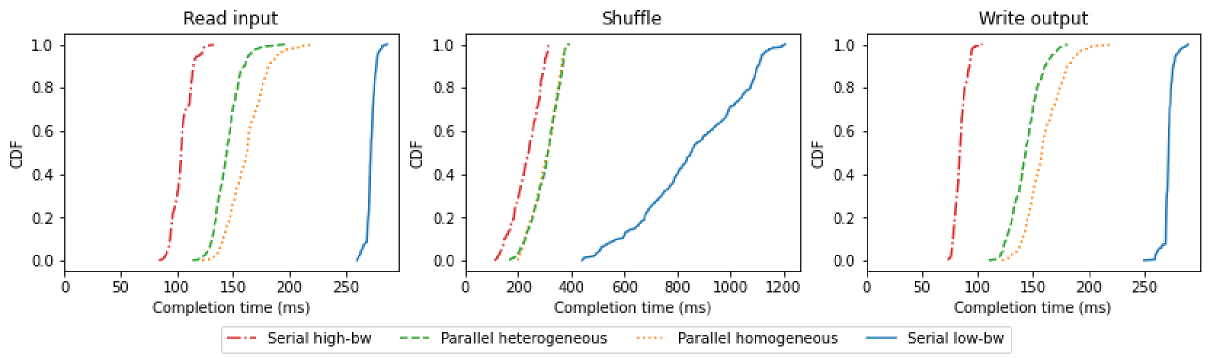


Figure 2.12. Simulated Hadoop-like workload per-worker completion time in each stage, single-path routing.

Figure 2.12 shows the distribution of observed network request completion times per worker, which is the total time it takes each mapper/reducer to load, shuffle, or write all the data it handles. We measure this at each mapper for the read input and shuffle stages and at each reducer for the write output stage. In stage 1 read input, as there are fewer hosts sending traffic, flows in parallel networks have less chance of colliding with other flows; thus they observe reduced overall worker completion time. Furthermore, flows in parallel heterogeneous networks can utilize shorter paths in sparse traffic settings and achieve lower flow completion times. In stage 2 shuffle, the serial low-bandwidth network suffers more from condensed traffic and has a long tail. The dense traffic allows flows to take close-to-full advantage of parallel networks and achieve closer to ideal serial high-bandwidth network performance. However, parallel heterogeneous networks do not exhibit additional advantages over their parallel homogeneous counterparts. This is because dense traffic like all-to-all in the shuffle makes more flows collide as they try to take advantage of the shorter paths. Furthermore, these flows are approximately the same size as the single-vs-multipath cutoff threshold. Increasing the flow sizes in high-speed (100/400 G) networks by multiplexing multiple block transfers per flow would increase the performance of this type of application. In stage 3 write output, each reducer writes to a random remote host designated to store that block; thus the traffic pattern looks like the inverse of stage 1 read input. Thus, we observe similar performance as in stage 1.

2.5.3 Published datacenter flow traces

We also used datacenter flow traces [87, 44, 7] to confirm our findings in Section 2.5.1 and Section 2.5.2. We plotted the flow size distributions of webserver/cache/Hadoop [87], datamining [44] and web search [7] applications in Figure 2.13.

For this experiment, we used a similar setup as in Section 2.5.1, with individual flow sizes drawn from the distributions in these traces, as shown in Figure 2.13. We set up four concurrent flows per host to saturate the network and each flow runs in a closed loop using single-path routing.

To start with, we look at the results on Jellyfish networks at 100/400 G for Datamining from VL2 [44] and Websearch from DCTCP [7], which represent small flows and large flows from these traces respectively. These results shown in Figure 2.14a and Figure 2.14b confirm

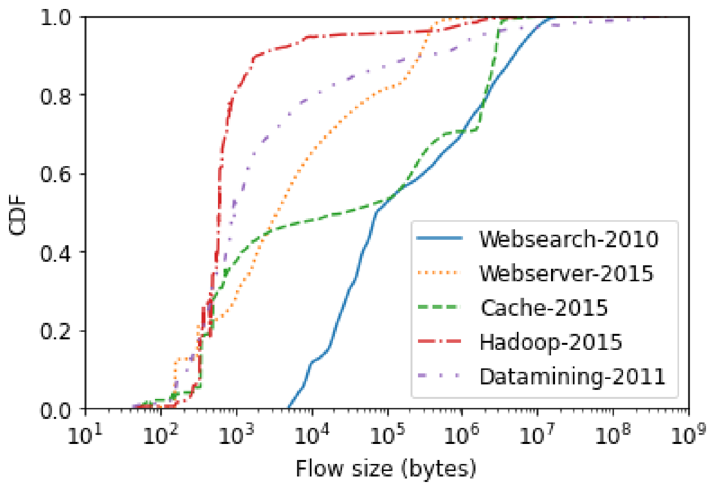
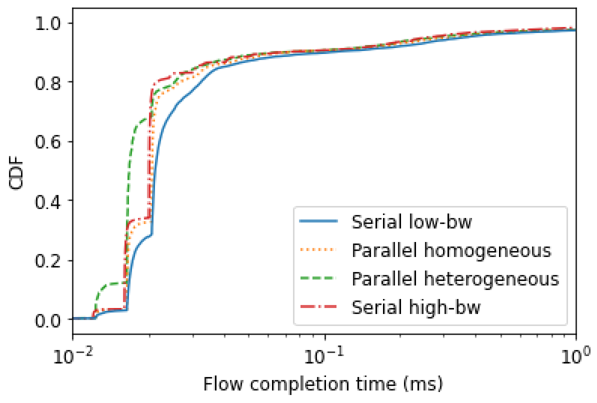
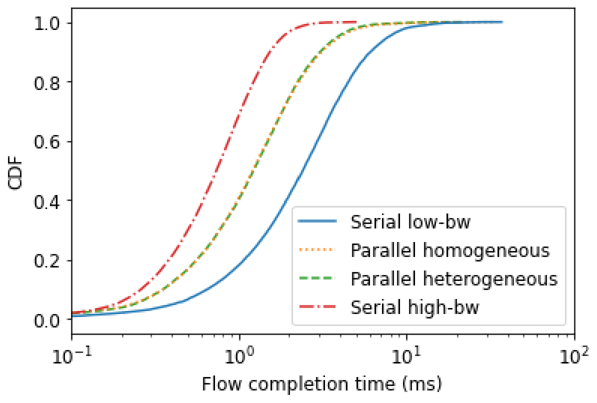


Figure 2.13. Flow size distribution of published DC flow traces from [87, 44, 7].



(a) Datamining trace [44] FCT



(b) Websearch trace [7] FCT

Figure 2.14. Flow completion time distributions of two published DC flow traces [44, 7].

our previous studies using synthetic traffic. In particular, Figure 2.14a shows that similar to the RPC-like application in Section 2.5.2, short flows like the Datamining traffic can achieve lower latency on P-Nets, especially parallel heterogeneous networks, by exploiting the lower average hop count and better tolerance of multiple concurrent flows. Figure 2.14b, on the other hand, shows that similar to the simulated shuffle traffic in Section 2.5.2, P-Net can achieve significant throughput improvement over serial low-bandwidth networks and closer to ideal high-throughput serial networks.

Next, if we look at the full comparison of all the traces across both Jellyfish and fat trees, we see that in general the fat tree results look similar to Jellyfish ones (except there are no parallel heterogeneous fat trees) and the smaller traces behave similarly to the datamining traffic.

Figures 2.15 - 2.19 show the CDF distribution of flow completion times (FCTs) of all five traces from Figure 2.13. We compared the result at both 10/40 G (top rows) and 100/400 G (bottom rows), as well as both fat tree topologies (left columns) and Jellyfish topologies (right columns).

As we can see from these results, at 10/40 G, P-Net can achieve lower latency on most flows, as it can provide better load balancing and accommodate multiple flows per host than serial networks, thus achieving close to ideal high-throughput network's performance. In addition, the path length advantage in parallel heterogeneous networks can sometimes provide further improvements over the parallel homogeneous ones. At 100/400 G, the path length advantage of heterogeneous 4×100 G P-Net shows its full strength, allowing certain short flows to achieve even lower latency than an ideal 400 G serial network.

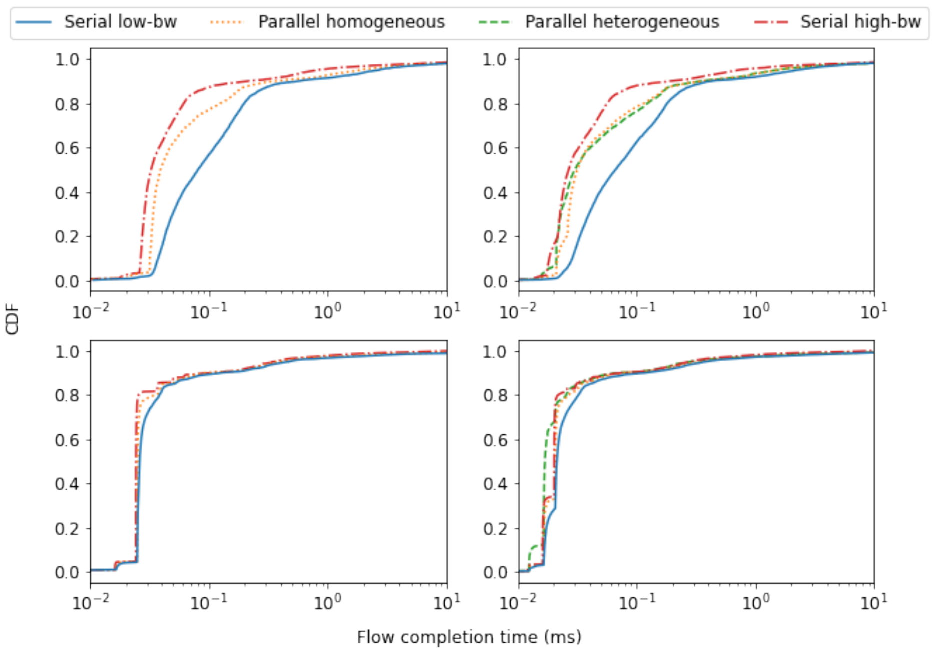


Figure 2.15. Flow completion time distribution based on Datamining traces from [44]. Top row plots are for 10/40 G and bottom row plots are for 100/400 G. Left column plots are for fat trees and right column plots are for Jellyfish.

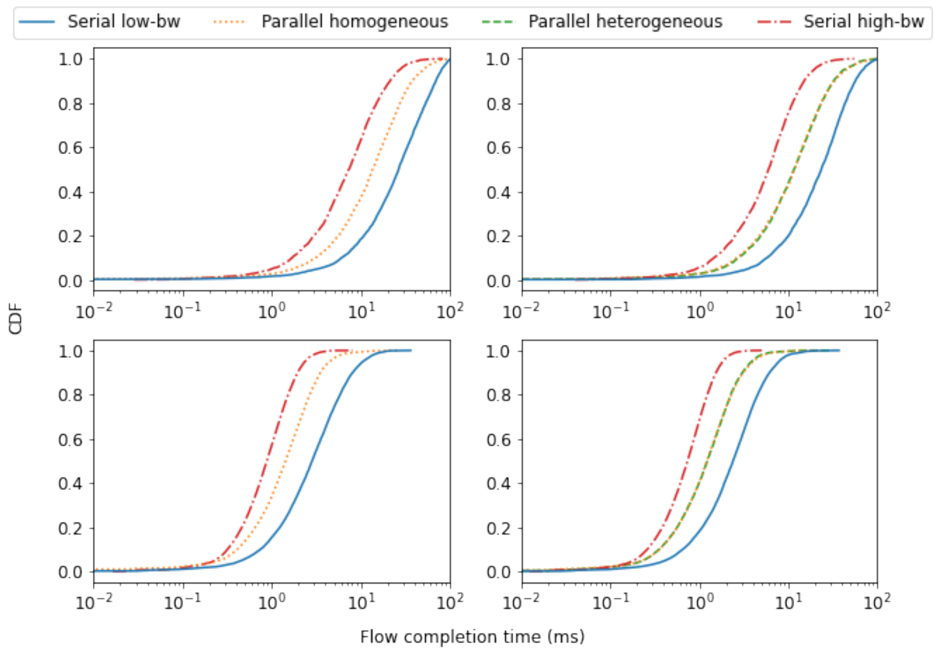


Figure 2.16. Flow completion time distribution based on Websearch traces from [7]. Top row plots are for 10/40 G and bottom row plots are for 100/400 G. Left column plots are for fat trees and right column plots are for Jellyfish.

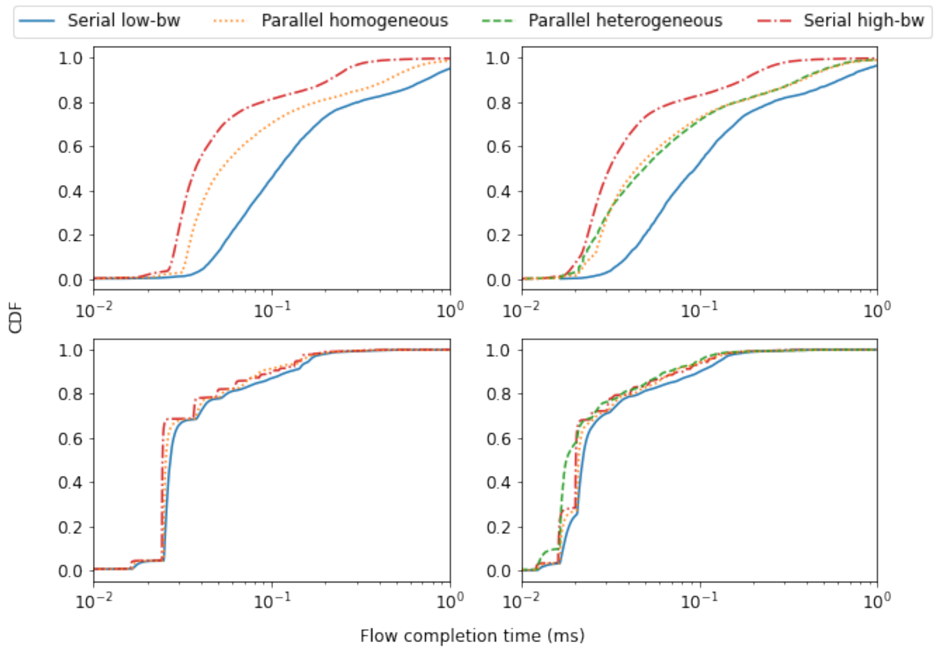


Figure 2.17. Flow completion time distribution based on webserver traces from [87]. Top row plots are for 10/40 G and bottom row plots are for 100/400 G. Left column plots are for fat trees and right column plots are for Jellyfish.

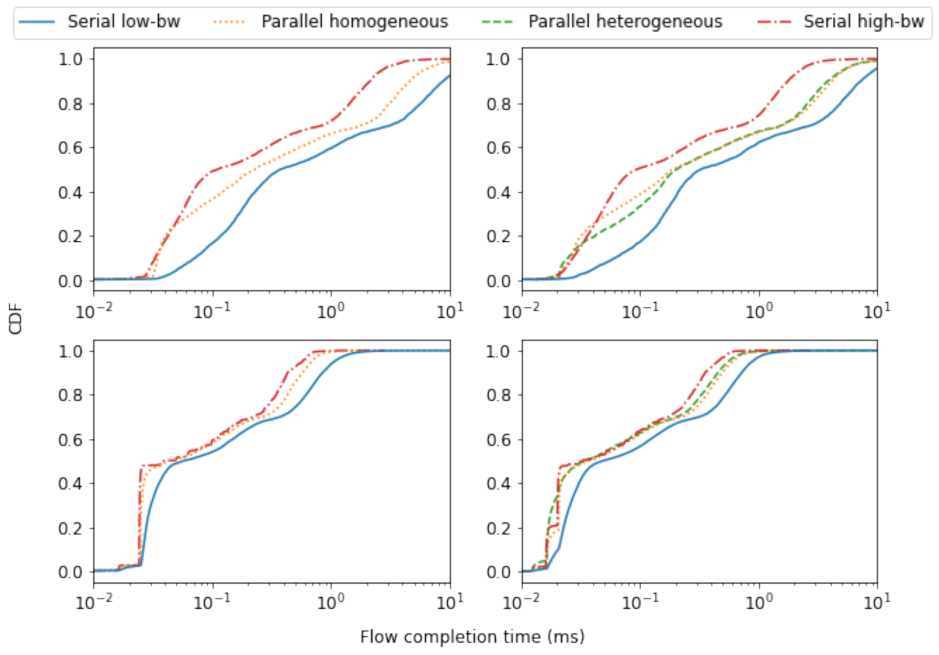


Figure 2.18. Flow completion time distribution based on cache traces from [87]. Top row plots are for 10/40 G and bottom row plots are for 100/400 G. Left column plots are for fat trees and right column plots are for Jellyfish.

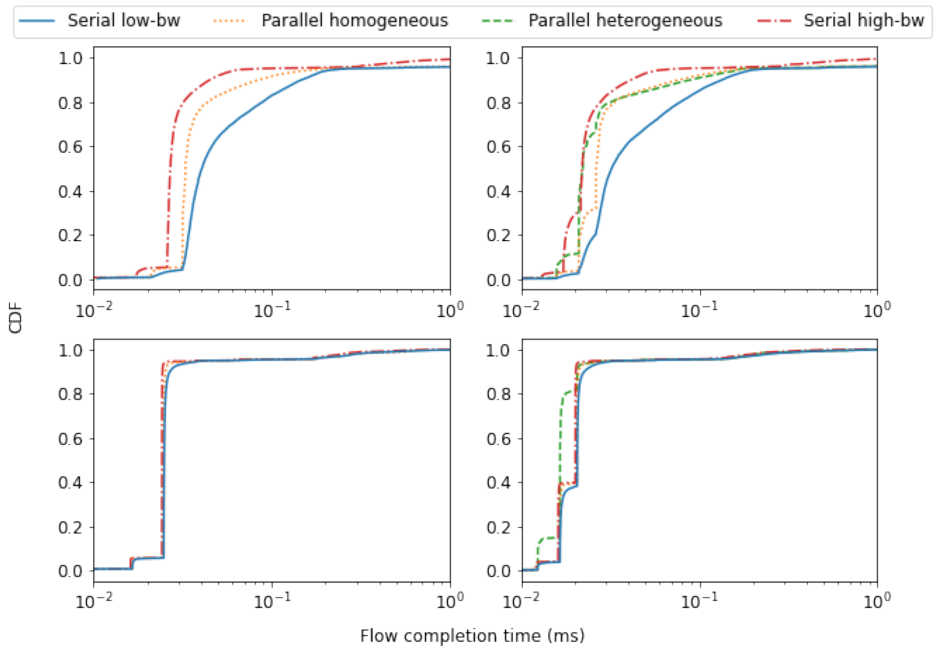


Figure 2.19. Flow completion time distribution based on Hadoop traces from [87]. Top row plots are for 10/40 G and bottom row plots are for 100/400 G. Left column plots are for fat trees and right column plots are for Jellyfish.

2.5.4 Fault tolerance

Another important benefit of P-Net is its better resiliency against network failures. Because P-Net consists of multiple dataplanes and thus more paths, it is less likely to lose all the shortest paths than serial networks with a single dataplane. This allows for more graceful performance degradation than traditional serial networks.

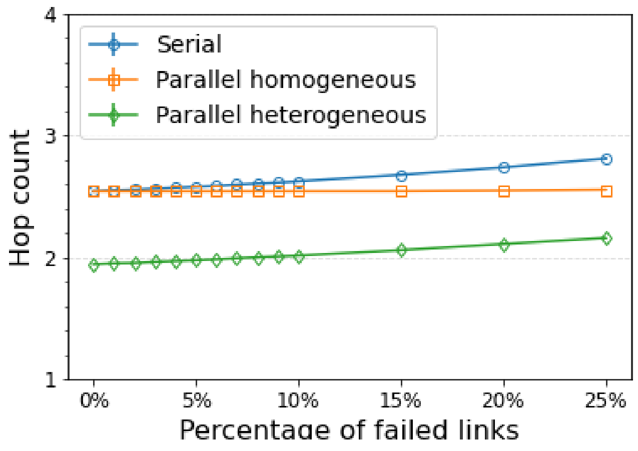


Figure 2.20. Average hop count across all src/dst pairs. Higher hop count indicates increased latency.

In Figure 2.20, we compare the impact on average hop count of all-pairs shortest paths in a set of Jellyfish networks with 686 hosts: serial, parallel homogeneous, and parallel heterogeneous. Note that serial low-bandwidth and serial high-bandwidth networks only differ in link speed, and are identical in hop count distribution; thus, we do not differentiate them. The two parallel Jellyfish networks both have 4× dataplanes. Link failures are random across the network.

We observe that serial networks lose short paths very fast, which leads to 22% more hops with 40% link failures, whereas parallel homogeneous networks only suffer by 3%. Parallel heterogeneous networks also lose the really short paths fast; thus, their advantage diminishes quickly as more links fail, but they still outperform the other two types. We note that expander networks like Jellyfish are already highly resilient to failures [94], but P-Nets further improve upon it.

In reality, this can be extremely helpful when operators deploy expander networks in some dataplanes of P-Net to support low-latency traffic like web search. The path length advantage of these expander networks, combined with the high failure resiliency of P-Net, can provide the lowest latency even in case of multiple network failures. Furthermore, P-Nets' rack-level network redundancy removes a major single point of failure in today's datacenter networks [66].

2.6 Practical Considerations

P-Net proposes a new networking solution by deploying multiple dataplanes, but operators may find it difficult to deploy multiple copies of switches, cables, NICs, etc. Here, we discuss some of the optimizations to alleviate these problems.

2.6.1 Network deployment

First of all, P-Net configures switches for higher radix and lower per-port speed, and uses multiple sets of switches and cables to achieve equivalent total bandwidth of a traditional network. This comes at a cost of more distinct elements, e.g. switch boxes and fiber cables. Fortunately, modern networking and deployment solutions can greatly reduce redundant efforts. The top and middle part of Figure 2.21 shows the deployment optimization for P-Nets.

For the aggregation/core layer, we use optical patch panels and optical circuit switches (OCSes) to simplify wiring. We borrow the idea of patch panels from [92] and [120], which has been shown to 1) significantly reduce wiring complexity by operating only on the patch panels, 2) leave room for more aggregation layers, and 3) allow easier reconfiguration and expansion. More importantly in P-Net, this enables us to "hide" the heterogeneity in heterogeneous P-Nets, as we show next in Section 2.6.2. OCSes like Calient/Palomar and rotor switches can provide additional improvements over patch panels by replacing the back-side wirings with either software [89, 81] or pre-etched gratings [31]. Furthermore, by adopting optical switching, we can eliminate transceivers in the network core. This has been shown to be a key scaling mechanism into Terabit Ethernet, as high-speed packet switches and transceivers consume extremely high power [93, 65,

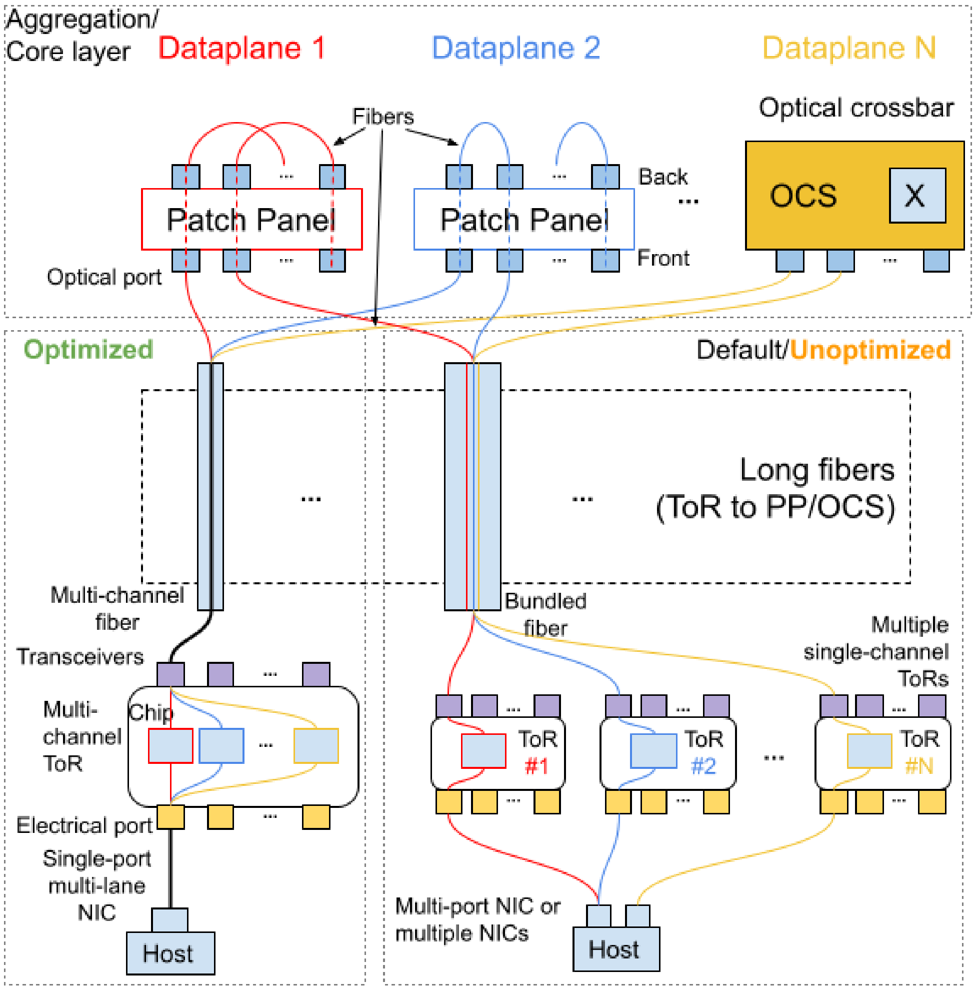


Figure 2.21. Deployment optimization of P-Net.

39].

For switches, by default, P-Nets use separate ToR switches and fiber cables to improve redundancy, as shown in the right dotted box. Each color represents one dataplane, with its own fibers connecting separate ToR switches and the aggregation/core layer on top. This provides the highest redundancy, at the cost of increased wiring efforts.

P-Nets can also take advantage of modern multi-channel Ethernet technologies [9] and coalesce links from multiple dataplanes into a single physical cable, as shown in the left dotted box. We can deploy 4×100 G P-Nets using the four 100 G channels in 400 G Ethernet cables [9, 76]. We split long-running fibers at the end and connect individual channels to different patch panels or OCSes. On the switch side, modern multi-channel switches already split these channels at the chip level, as in scale-out and chassis networks. P-Nets can adopt the same design, but with a flattened layer of chips inside each switch box to reduce chip count and power density, while still providing connectivity to multiple dataplanes.

Last but not least, by using multiple dataplanes, P-Net allows operators to upgrade one dataplane at a time, without bringing down the entire network. Furthermore, software-controlled OCSes together with the incremental expansion support of expander-based networks means operators can more easily scale up their network.

2.6.2 "Hiding" heterogeneity

The concept of heterogeneous P-Nets may worry operators and network designers, as deploying and managing *N* different networks seem very challenging, but here we present ways to reduce that heterogeneity to the minimal level.

As we discussed above, patch panels and OCSes allow operators to *localize* the heterogeneity across dataplanes to a central location. This means that the long-running fibers and per-rack layout can be kept exactly the same, keeping deployment complexity across the datacenter floor low. Furthermore, by using software-controlled OCSes or pre-etched gratings that encode the connectivity, operators can completely hide the hardware heterogeneity in P-Nets.

2.6.3 End host/NIC support

For end hosts, we can further reduce the rack-level wiring complexity by using single-port-multi-channel NICs like the HPE 4×25 Gb 1-port 620QSFP28 adapter [30] and FPGA-based NICs like Corundum [35]. The former provides four separate 25 G channels to a server using a single 100 G port at an extremely affordable price, and the latter provides similar multi-channel functionality with more flexibility. Operators can balance between ToR redundancy and cost by varying the number of physical uplinks.

2.6.4 End-host burst capacity

We note that the design of P-Net limits the per-port bandwidth to $1/N\times$ of the state of the art, which may hurt the performance of high-throughput applications like GPUs, TPUs and other accelerators. However, first, we observe that very few datacenter applications can saturate a single 100+G link by itself [87, 20]. Second, multiple flows per host can be effectively spread across the parallel dataplanes in P-Nets. Lastly, these high-throughput applications often use specialized interconnect solutions anyway, due to their unique communication patterns and requirements, as shown in Google's 2D and 3D Torus networks used in their TPU clusters [59, 65].

2.6.5 Incast traffic

For incast scenarios, P-Net can spread the traffic across separate dataplanes to alleviate congestion in the network, but careful coordination is still needed to avoid overrunning end host NIC buffers. We defer this to future studies that might involve incast-aware transports like DCTCP [7] or NDP [51].

2.7 Future work and opportunities

Monitoring and diagnostics:

P-Net's adoption of multiple dataplanes brings management and diagnostic challenges, since each dataplane is logically separate and often belongs to a different control domain. Existing systems will need to merge flow statistics from multiple dataplanes to accurately describe the network state and troubleshoot issues.

P-Net with different topology types:

Although we discussed parallel heterogeneous networks, all dataplanes have the same type of topology. Another type of parallel heterogeneous network can consist of entirely different topologies across the dataplanes. For example, operators can deploy a combination of expander-based topologies and fat trees to handle both low-latency traffic and Hadoop-like data-intensive workloads. The major difficulty then becomes managing a mixture of network topologies within a datacenter.

Performance isolation:

Because P-Net has multiple isolated dataplanes, operators can assign different traffic classes to different dataplanes to achieve performance isolation. For example, user-facing frontend traffic can be assigned to one dataplane, and background data analysis traffic can be assigned to another. Traffic from different tenants can also be assigned to different dataplanes to avoid interference. This strict performance isolation opens up new opportunities for control-plane solutions like pFabric [8], EyeQ [58] and pHost [36], which attempts to support a mixture of elephants and mice traffic on a single fabric.

2.8 Related work

Parallel dataplane networks have begun to be adopted by the industry as we move towards higher link speeds. Facebook [12] and LinkedIn [119] have built parallel fat trees to achieve equivalent 400 G speed in the network core using 4×100 G dataplanes. Compared with P-Nets, their topologies are homogeneous, and end hosts are limited to using one of the dataplanes via ECMP. This leads to lower operational costs than chassis networks, but they do not have the additional benefits of heterogeneous P-Nets.

In the research community, our previous workshop paper [74] also explored parallel networks design, but was limited to parallel fat trees. This chapter extends on it by 1) further exploring parallel heterogeneous networks, in which flows can pick the dataplane(s) with the shortest hops to achieve lower latency and higher total throughput, 2) studying and answering practical issues of heterogeneous P-Net deployment, and 3) empirically evaluating the performance of several datacenter network traffic patterns in P-Nets.

P-Net is not the first to use multiple uplinks in datacenter environment. GRIN [5] and

Subways [66] are two proposals that utilize multiple NICs/ports per end host to improve network performance. GRIN [5] connects servers in the same or adjacent racks together using the additional ports, allowing each end host to opportunistically send/receive traffic via its paired neighbor. It can boost end-host burst capacity at the cost of additional cross-rack wirings. Subways [66] takes a slightly different approach by connecting end hosts directly to the neighboring rack's ToR switch. It uses adaptive load balancing to lower congestion, and similar to P-Net, provides better fault tolerance via redundant ToR switches.

Stardust [124] is another datacenter network design that separates a single link into multiple lower-speed links. It proposed a new Clos-based network design that uses cell-based forwarding, simplified routing, and redesigned hardware to improve the performance and power savings of datacenter networks, but is still a serial network with one dataplane. P-Net focuses on an orthogonal problem of datacenter network design, which is whether and how we can use multiple dataplanes to support growing application demand.

2.9 Summary

Evolving datacenter networks to meet the bandwidth demands of next-generation systems is a challenge, in part due to limited scaling of packet switch chip speed. Careful packaging of switch chips within chassis helps reach part of the scaling goal, but becomes cost- and energy-prohibitive at future link speeds. In this chapter, we explored the space of parallel dataplane networks (P-Net) to add a new degree of scaling to network designs, which helps bridge the gap between fast-growing application demand and current-generation hardware capacity. We showed that naive routing in P-Nets leads to suboptimal performance, and that with careful path selection and multipath transport, we can scale beyond existing switch chip bandwidth limits and achieve low latency. With careful practical deployment considerations, we showed that P-Net can be deployed in existing datacenter networks using existing commodity Ethernet switches. By eschewing the idea of a single network, P-Net provides an alternative to the high-cost and high-power chassis architecture that achieves $2-3\times$ reduction in hops, chip count and power consumption while doubling the bandwidth.

P-Net is a new datacenter network design that shows improved performance, lower cost and additional power savings than existing chassis-based data center networks. We believe that P-Net can be a stepping stone towards more scalable and efficient datacenter networks.

In the next chapter, we will focus on the carbon emissions aspect of data center sustainability, and explore how to achieve the carbon emissions reduction of data centers by using renewable energy sources and carbon-aware workload migration.

2.10 Acknowledgements

Chapter 2 is largely a reprint of the material as it appears in the Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '22), Yibo Guo, William M. Mellette, Alex C. Snoeren, and George Porter, Association for Computing Machinery, 2022. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Carbon-aware workload scheduling: Overview

In addition to traditional energy efficiency goals, the carbon footprint of data centers has become a growing and more urgent concern, as shown in both Google's and Microsoft's 2030 goal of achieving carbon neutrality [42, 19].

Recent studies have shown that in addition to consuming a significant amount of energy, data centers are also responsible for a large amount of carbon emissions [60] at 2%, or roughly on par with global aviation industry, and the numbers are expected to grow in the future [38] due to the rise of AI/ML applications and their higher energy consumption [26]. This raises concerns about the environmental impact of data centers and the need to build more sustainable data centers [52, 4]. In this chapter, I will discuss how to improve the carbon emissions of data centers by using renewable energy sources and how to better adopt these renewables through carbon-aware workload migration over the more energy-efficient wide-area network (WAN).

In Section 3.1, I will first discuss the challenges of adopting renewable energy like solar across a series of globally distributed data centers with different carbon intensity distribution, and how we can transform data center operations to better utilize these growing renewable energy sources. In Section 3.2, I will then discuss the design and implementation of a carbon-aware workload scheduling platform that can reduce the carbon emissions of data centers by moving workloads to data centers with lower carbon intensity. The high-level insight is that instead of moving renewable energy via the power grid, where we observe not only high transmission losses but more importantly insufficient capacity, we can move the workloads over the WAN, where we have the same coverage as the power grid but with cheaper marginal cost and much lower energy consumption.

3.1 Harnessing solar energy via globally-distributed data centers

Data centers have begun adapting to renewable energy sources like wind and solar in recent years to reduce operational carbon emissions. As the percentage of renewables increases in the electricity grid, time-shifting and space-shifting techniques have been employed to *gradually* align data center power usage with variable availability of renewable energy.

In this work, we propose a future computing architecture where data centers around the globe rely primarily on solar energy. To cope with the variation of solar power, we move workloads away from locations at night and instead run them at locations in daytime. We present the initial design of this architecture, and raise several important questions on how to transition from today's mixed-energy-powered data centers to future solar-powered data centers.

3.1.1 Introduction

Data center energy usage and carbon footprint have been exploding rapidly thanks to the increasing computing demands from artificial intelligence and machine learning, and the move to the cloud [99]. To reduce their operational carbon emissions, operators have begun exploring ways to better adopt low-carbon renewable energy sources, like solar and wind. This is a challenging problem because these renewables often have variable availability, and thus there are always times of day and locations where there's little renewable energy available.

To address this variable availability issues, several studies have looked into time-shifting and space-shifting techniques to align data center power usage with the availability of renewables [41, 83, 111, 6]. Because data centers today are powered by a mix of low-carbon renewables and high-carbon energy sources, these studies have been focused on how to *gradually* change existing data centers in response to growing renewables, but ultimately data centers need to be powered primarily by these time- and space-varying renewables.

In this section, we explore a future computing architecture where a series of globally-distributed data centers that are powered primarily by solar. In order to reduce the use of stable power that's like high-carbon, we aggressively move workloads to locations with ample solar power to reduce. Because of the fairly predictable nature of solar power, we can move workloads to "chase" the sun, as shown in Figure 3.1. This can be achieved by reducing workloads at locations where the sun is about to set and starting new workloads at locations in their early morning hours.

We recognize that this design creates additional challenges in workload scheduling and data center capacity, but future renewable-based society may require more computing to be powered by these variable renewables.

Thus in this work, we want to quantitatively evaluate the cost of this solar-powered data center architecture, and compare it with existing data centers that do not move workloads around aggressively. In particular, we emphasize on these questions:

- how to schedule workloads around these globally-distributed solar-powered data centers?
- what's the impact of moving to this renewable-based architecture for each type of workload?
- what's the additional hardware cost to support varying amounts of workloads in this architecture?
- what's the minimum stable power required to support workloads that cannot be moved?

3.1.2 Globally-distributed solar-powered data centers

Our proposed architecture consists of a series of globally-distributed data centers across multiple time zones, as shown in Figure 3.1. This figure shows the daytime and nighttime on

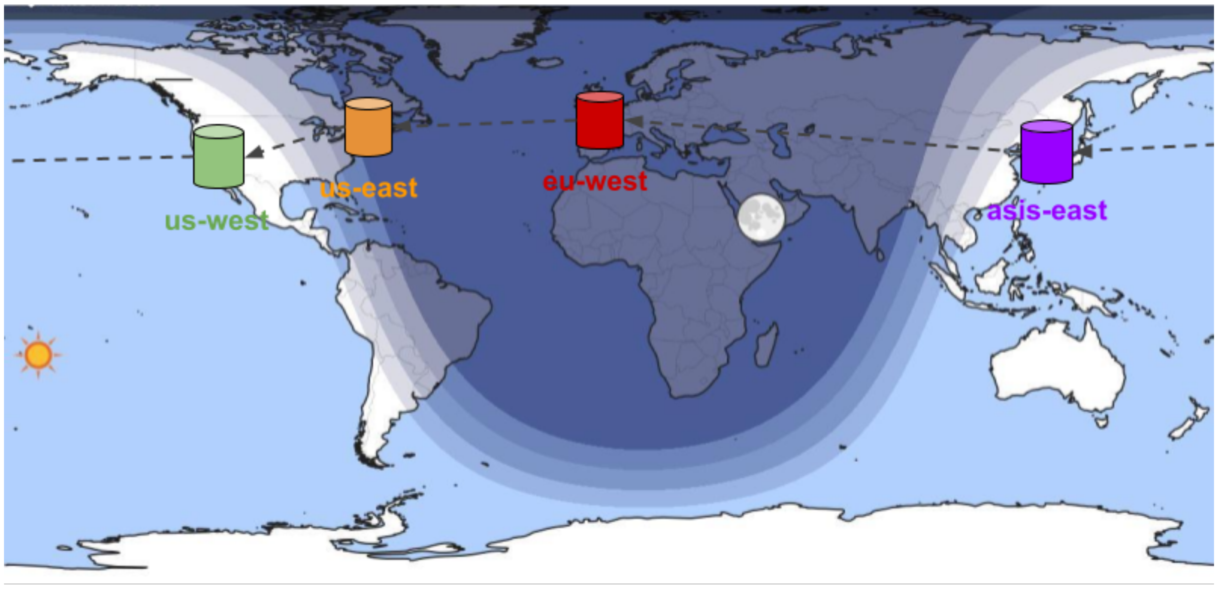


Figure 3.1. Coarse-grain solar energy availability in globally-distributed solar-powered data centers.

a global scale, which is a coarse-grain approximation of solar energy availability. A few data centers across different time zones are shown here, which represents several major public cloud locations.

Note that at any given time, there will always be roughly half of the world lit by the sun, and thus up to half of the data centers can directly use solar energy. Our goal is to run most workloads at these locations to maximize solar energy usage and move workloads away from locations at their nighttime. As the earth rotates and the daytime region shifts westbound, we can move workloads along with the sun.

In this particular case, we can see that `us-west` and `asia-east` are in daytime and `us-east` and `eu-west` are in nighttime. Thus it's desired to run more workloads in these locations now. But as time goes by, we can see that `us-west` will start to lose solar power as the sun sets. Thus we need to stop assigning new workloads or stop existing workloads at `us-west`, and instead, move or run new workloads at data centers to its west, as shown in the arrows in Figure 3.1.

3.1.3 Challenges

Although solar-powered data centers can greatly reduce its operational carbon footprints, this new architecture poses several major challenges to how we run data centers:

3.1.4 Scheduling challenges

Careful scheduling of workloads across data centers is needed.

Given the distribution of data centers around the globe, their solar availability may vary and thus we need to schedule workload movement so that each data center "hands off" its work when it starts to lose solar power. We also need to consider the maximum capacity of each data center, the available wide-area network (WAN) bandwidth between data centers and the time needed to complete the hand-off (Section 3.1.5).

Not all workloads can be moved easily.

Existing commercial time-shifting solutions are often limited to internal flexible workloads [83] to avoid service level objective (SLO) violation, and space-shifting solutions are limited to short distances [6], due to their high impact on latency [37]. However, in future solar-powered data centers, such rigid workloads can incur too much carbon emission. Thus it is desirable to move as many workloads to "follow the sun" as possible, and we need to quantitatively measure the impact of moving each type of workload to determine the feasibility. This includes the overhead of setting up a new runtime, duplicating the input data, copying input/output data among data centers, and for online services, higher latency and WAN usage.

3.1.5 Capacity challenges

Additional WAN capacity is needed to absorb migration traffic.

As shown in [6], even moving virtual machines between three data centers within a small region can result in up to 40% of existing WAN capacity. To support continuous workload movement around the globe, we need to measure the additional WAN bandwidth needed to move these workloads, and apply techniques like pause-and-restart [111], careful selection of replication sites and planned replication ahead of time. Furthermore, this can impact workload scheduling decisions, as workloads depending on large amounts of data need to wait longer for data transfers to start running.

Extra hardware capacity is needed to absorb larger power variations of renewable energy sources.

Due to the variation of available renewable power, future data centers will likely have higher fluctuation in loads across times of day [4]. This will be especially true for solar-powered data centers where there's a clear daily pattern. To accommodate such power fluctuation, operators can either increase processor frequency (at higher energy costs, but less of an issue when using abundant solar energy) or purchase additional hardware, which will increase the embodied carbon footprint of data centers [4]. Thus, careful consideration must be taken to balance the reduction in operational carbon emissions and the increase in embodied carbon footprint, and it's important to determine the scale of power fluctuation in this solar-powered data center model.

3.1.6 Summary

In this section, we explore a future computing architecture where data centers around the globe are primarily powered by solar, and we shift workloads to "chase" the sun to reduce the use of other high-carbon energy sources.

Although this design may introduce additional scheduling and capacity challenges, it is a necessary step to reduce the operational carbon footprint of data centers. We lay out the major challenges to transition today's data centers to this future architecture, and the next steps of further quantitative analysis to determine the cost and feasibility of such approach.

Based on these insights, we first show the high-level design of a carbon-aware workload scheduling platform in the next section. In particular, we propose a inter-datacenter workload scheduling and placement framework across multiple data centers around the world, and by examining both carbon savings and workload migration carbon costs, we obtain a key insight on how to balance between carbon savings and the associated migration overhead, which is crucial to the success of this platform.

3.2 Carbon-aware inter-datacenter workload scheduling and placement

Data centers have begun adapting to renewable energy sources in recent years to reduce their operational carbon emissions. As the percentage of renewables increases in the electricity grid, time-shifting techniques have been employed to gradually align data center power usage with the variable availability of renewables [83, 111]. But there is a limit on how much we can move workloads around in time within a data center.

In this work, we study another dimension to solve the renewable intermittency problem, i.e. space-shifting workloads via geographical workload migration. We propose an inter-datacenter workload scheduling and placement system that considers both carbon cleanness and migration overhead. We define a new metric that quantifies the cost of workload migration and use it to prioritize moving jobs with low migration cost. Our initial prototyping on Kubernetes based on three regions shows promising solutions using the built-in scheduling policies and external carbon signals. Future integration with live carbon data prediction is needed for online evaluation.

3.2.1 Introduction

Data center energy usage and its resulting carbon footprint have been growing rapidly thanks to AI/ML applications and cloud migration [99]. Aligning datacenter power demands to intermittent renewables is a challenge, and several studies have examined time-shifting workloads to match the available low-carbon power [83, 111]. But time-shifting with a single data center limits the carbon saving potentials because: 1) many jobs cannot wait overnight for the next solar window, 2) there is not enough demand within a data center to take advantage of the high solar peak, and 3) time-shifting cannot address the geographical imbalance of renewables, which is also prevalent in the grid. Alternatively, Agarwal et al. explored space-shifting in workload-agnostic settings [6], but it was limited to a small region due to high moving costs.

We believe that to really embrace migration as a solution to this problem, we must acknowledge that *different jobs vary in their cost to move*. A simple example is code compilation vs log analysis. The former is CPU-heavy and can be easily moved, whereas the latter is data-heavy and will require a large data transfer. Thus, one of our goals in this work is to differentiate different classes of applications and quantify the cost of moving each type of application.

In this work, we propose a comprehensive inter-datacenter workload scheduling and placement system that optimizes for both carbon footprint and migration cost. At the high level, we employ a two-level scheduling system that 1) adjusts resource footprint across data centers based on their carbon cleanness, and 2) assigns individual jobs to their optimal locations while considering both carbon savings and migration cost. We will now discuss the system design, explain migration cost analysis, and end with our evaluation plan.

3.2.2 Multi-datacenter carbon-aware scheduling

At the core of our system is a two-layer scheduler that accounts for both carbon emission reduction and migration cost. Figure 3.2 shows the architecture and main components.

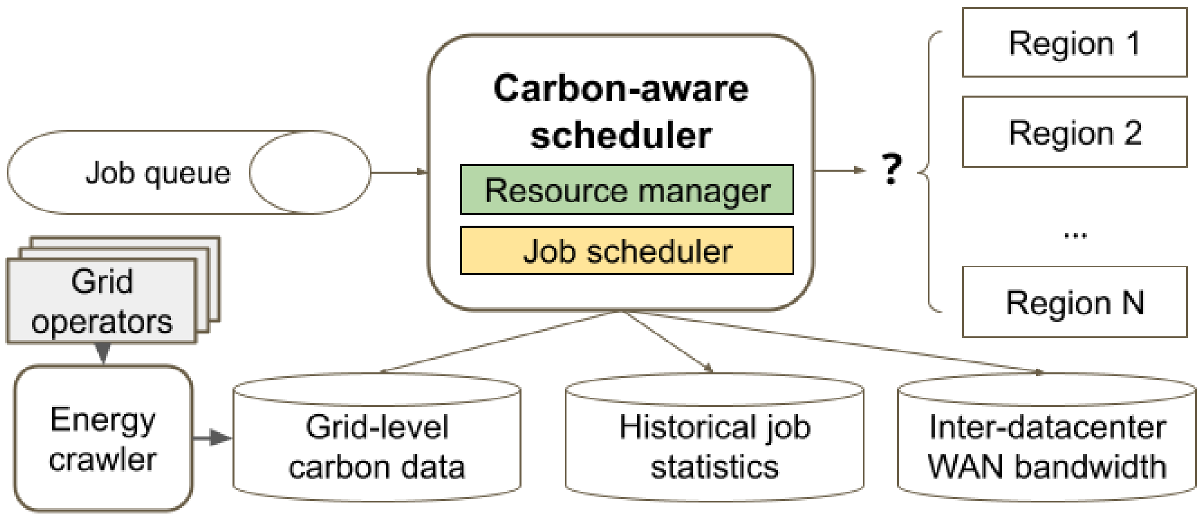


Figure 3.2. A geo-distributed carbon-aware scheduler architecture.

The top-layer `resource manager` adjusts the available resources in each region, based on the carbon cleanness and current utilization. If there are more workloads, it prioritizes the region(s) with low-carbon power; and when there are fewer workloads, it reduces the resource footprint in high-carbon regions. This operates at a lower frequency, e.g. 15 min. The bottom-layer `job scheduler` makes real-time decisions on where to run a job, by considering available resources, carbon cleanness, and migration cost. The goal is to avoid moving jobs with high migration costs that negate the carbon savings.

The carbon emission reduction is calculated as the compute energy usage times the carbon intensity difference between two regions. We measure the compute energy using profiling tools like `Intel RAPL`, or calculate it as CPU time $\times$ TDP. Carbon intensity is a well-defined metric that measures the carbon emissions per unit of electricity (`gCO2/kWh`). We calculate this time series data for each region based on its local grid energy supply mix, which we record using a data crawler.

For migration cost, we focus on the energy needed to perform the data transfer, which is the bulk of overhead from an energy perspective. We collect historic job execution information like job run time and input/output data size, to determine its compute energy usage and migration cost. Further, prior studies have shown that WAN bandwidth can vary significantly across region pairs and times of the day. Thus, we also keep track of this information using past transfers and periodic probes.

3.2.3 Balancing migration cost with carbon savings

To balance between carbon savings and migration cost, we calculate how much overhead in terms of energy consumption we are adding by moving a workload ($+X\%$), and compare it with the carbon savings of such movement ($-Y\%$). If X is on par with Y, then it's not worthwhile to move this workload, but if $X \ll Y$, this means that we can achieve carbon savings with relatively negligible overhead.

This intuition incentivizes us to define this new metric to guide migration decisions, which is the ratio of a job's compute energy usage and input/output data size. More formally, we define:

$$\textbf{compute-to-data-size ratio} = \frac{\text{Compute energy usage}}{\text{Data size}}$$

Intuitively, a high compute-to-data-size ratio means that the job is more CPU-heavy. Thus, moving it will recur less data movement per unit of compute energy usage, or carbon savings. This allows us to prioritize workloads with high compute-to-data-size ratios, as moving these jobs can move more energy consumption and thus achieve more carbon savings per byte of data movement.

We performed an analysis of several common workloads and the results are shown in Table 3.1. Note that this alone does not determine the actual migration cost, because data transfer time also depends on the available WAN bandwidth. However, this metric is a great indicator of how easily a job or a type of job can be moved. We combine this singular metric with WAN bandwidth data to make real-time decisions in our scheduling pipeline.

Table 3.1. Compute-to-data-size ratio of profiled workloads.

Workload	Compute-to-data-size ratio (kJ/GB)
Compile Linux	76.42
Video effect (grayscale, h.265)	96.8
Video effect (grayscale, h.264)	11.53
Video transcoding (h.264 $\rightarrow$ h.265)	19.54
Video resizing (4k $\rightarrow$ 1080p, h.264)	1.41
Compression (gzip)	0.47

3.2.4 Prototype evaluation on Kubernetes

We prototyped our design on Nautilus, a Kubernetes research platform that supports three US regions: `us-west`, `us-central` and `us-east`. This is an ideal platform because Nautilus allows us to easily schedule workloads in any region and arrange data transfers. Nautilus also contains many batch jobs that are delay-tolerant.

We integrated the carbon-aware scheduling policies into Kubernetes using custom scheduling rules prioritizing low-carbon regions, which is queried from a carbon-aware scheduling API. Initial integration with the S3 storage system allows us to transparently move the data associated with the job to the target region, demonstrating the feasibility of such implementation on Kubernetes. Future work is needed however to integrate with carbon data prediction systems like [69] for online evaluation.

3.2.5 Summary

In this section, we present a carbon-aware workload scheduling and placement system that moves workloads across data centers transparently to achieve a lower carbon footprint while managing migration overhead. We build an initial network carbon model, perform migration cost analyses based on the model, and define a new metric to guide migration decisions and avoid high-cost migrations. We prototype our system on Kubernetes and demonstrate its feasibility in a practical multi-region setting.

3.3 Carbon-aware workload scheduling: Summary

In this chapter, I discussed the high-level landscape of carbon-aware computing in data centers and the associated challenges.

In Section 3.1, I first discussed the trend in renewable energy deployment, and how we can transform data center operations to better utilize growing renewable energy sources like solar power. I then discussed in Section 3.2 the design and implementation of a carbon-aware workload scheduling platform, that can reduce the carbon emissions of data centers by moving workloads to data centers with lower carbon intensity. Our initial modeling shows a critical insight: picking the right type of workload and balancing the carbon savings and the associated migration overhead is crucial to the success of this platform.

Based on this insight, I will focus on the wide-area network (WAN) aspect of carbon-aware space-shifting in the next chapter. Space-shifting aims at moving the workload to data centers with lower carbon intensity in order to lower the computational carbon emissions. The hypothesis is that space-shifting can be more effective than time-shifting in many cases, due to the widely distributed nature of renewable energy sources and the varying carbon intensity of the electricity grid across geographical regions. However, as shown in Section 3.2, space-shifting is also limited by the carbon cost of moving the workload across data centers. In the next chapter, I will study the effect of the network on carbon-ware space-shifting in depth. In particular, I will propose a detailed carbon accounting method for workload migration across data centers over the WAN, and evaluate the carbon savings of space-shifting in a practical setting.

3.4 Acknowledgements

Chapter 3 contains material from two different sources co-authored by the dissertation author. Each source is used in a different part of the chapter.

Section 3.1 contains material that appeared in the 1st Workshop on NetZero Carbon Computing (NetZero '23), Yibo Guo and George Porter, 2023. The dissertation author was an investigator and co-author of this material.

Section 3.2 contains material that appeared in the Poster session of 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI '23), Yibo Guo and George Porter, USENIX Association, 2023. The dissertation author was an investigator and co-author of this material.

Chapter 4

Carbon-aware workload scheduling: The effect of network

Now that I have discussed the high-level implications of carbon-aware workload scheduling and the importance of the network in space-shifting, I will move onto the details on how to quantify the network carbon cost of moving workloads across data centers. In this chapter, I will discuss how we improve upon the existing space-shifting approaches to account for the carbon cost of the WAN, and how we compare against alternative time-shifting solutions in achieving carbon savings.

Organizations today are increasingly offloading their workloads to cloud platforms. For workloads with relaxed deadlines, this presents an opportunity to reduce the total carbon footprint of these computations by moving workloads to datacenters with access to low-carbon power. Recently published results [97] have shown that the carbon footprint of the wide-area network (WAN) can be a significant share of the total carbon output of executing the workload itself, and so careful selection of the time and place where these computations are offloaded is critical. In this chapter, we propose an approach to geographic workload migration that uses high-fidelity maps of physical Internet infrastructure to better estimate the carbon costs of WAN transfers. Our findings show that space-shifting workloads can achieve much higher carbon savings than time-shifting alone, if accurate estimates of WAN carbon costs are taken into account.

4.1 Introduction

Organizations have embraced cloud computing to a significant degree. In 2020, spending on cloud platforms grew by 35% to reach almost $130 billion, whereas spending on on-prem datacenters dropped by 6% to under $90 billion [1]. A major reason that organizations migrate their compute and data to these platforms is to achieve features and properties that would be difficult or cost-prohibitive for them to implement themselves [102, 14]. Migrating work to the cloud permits scaling up (or down) resources in response to changes in user behavior or need. Migrating data geographically can enable replication for improved availability and fault tolerance. And more recently, cloud adoption can help organizations meet climate-related goals such as limiting the carbon footprint of their compute needs.

Cloud providers are also using the flexibility of their platforms to reduce their own carbon footprints. Researchers have proposed [111], and Google has publicly shown [82], how to shift workloads with relaxed deadlines forwards or backwards in time to align with the availability of lower-carbon energy sources (such as wind and solar photovoltaic). The net result is delivering a lower carbon footprint for the same underlying computation. Given that low-carbon energy sources are heavily affected by environmental conditions (the sun in the case of solar, and wind speed in the case of wind power), there is significant potential in migrating data and workloads geographically to match compute demands with the status of energy grids and environmental conditions [37, 77, 57]. Already, geographic migration and replication is a key enabler of reducing wide-area latency [37, 114], better managing WAN bandwidth [6], or lowering monetary cost [90].

Any optimization task requires accurate information on the system's underlying resource utilization. Reducing the CPU requirements of a network-bound application is unlikely to deliver results in the same way that increasing the IO resources of a CPU-bound application will likely be unproductive. For geographic migration of cloud workloads, accurately estimating the carbon impact of the wide-area network (WAN) transfer is critical to reducing the overall carbon footprint of the computation. We observe that inaccuracies or omissions in estimating the carbon cost of the WAN lead to migration decisions that fail to achieve carbon optimization goals, and could even be entirely counterproductive. For example, as we'll show in Section 4.2.3, a network-heavy job can achieve 75% carbon savings without considering the network carbon cost, but if we account for the network transfer cost, we may not even want to migrate at all.

In this chapter, we show that previous approaches to estimating the carbon impact of WAN transfers underestimate the real impact by up to $2-3\times$ due to inaccurate understandings of the underlying routes, and we describe improvements to these approaches to better capture a higher fidelity picture of the carbon costs of WAN transfers. In particular, the contributions we claim in this chapter are:

- We improve upon state-of-the-art WAN carbon transfer estimation approaches based on traceroute data (Tabaeiaghdaei et al. [97]), as shown in Table 4.1 and Figure 4.2.
- We have built a dataset that maps components of WAN paths between cloud platforms to their underlying energy grids (called ISOs), and will make this publicly available for other researchers to build upon and to use to verify and replicate our network transfer cost results (see Section 4.4.4).
- We have analyzed algorithms for carbon-aware workload scheduling between geographically distributed data centers, quantitatively analyzing the potential additional benefit of space-shifting solutions over previously published time-shifting solutions.

4.2 Background and motivation

In this section, we first describe the high-level motivations of our work on carbon-aware workload migration, and then define the terms and definitions that we use throughout the chapter; finally we show a concrete example that highlights some of the factors impacting a successful workload migration.

4.2.1 Motivations

We first describe the motivations for general carbon-aware computing in Section 4.2.1, and then the specific reason advocating for space-shifting in Section 4.2.1.

Increasing renewable deployment

A key trend in carbon-aware computing is to adapt to the availability of renewable energy, which varies in both time and space domain [83, 111, 122, 82]. This means that some data centers may have access to a lot of low-carbon renewables (e.g. hydro- or nuclear-based locations) and others may have none (e.g. coal- or gas-based locations); or they may only have access to low-carbon during certain times of day or certain months of the year (e.g. solar- or wind-based locations) [82].

As the amount of renewable deployment continues to grow, existing grid infrastructure already lacks the capacity to store (in battery) or move (over long-distance grid lines) this excess renewable energy, due to various practical reasons like technological limitations, regulations and monetary costs. This leads to an increasing amount of curtailment, or wasted renewable energy, especially in solar powered regions like California [13].

As data centers are now accounting for an increasing and non-negligible share of global electricity usage [100, 99], researchers and data center operators have looked into how data centers can help alleviate this problem by shifting their power demand in time [82, 83, 111] and space [77, 122], which is a key mechanism to achieve the 24/7 carbon-free climate goals of many major companies.

Additionally, such a trend has led to closer integration of renewable power and data centers, as shown in both academic [6] and industry [61] worlds. This further demands cost-efficient carbon-aware solutions in not only time-shifting, but more importantly space-shifting.

Lower cost in WAN vs power grid

A key infrastructure that we leverage in carbon-aware space-shifting is the wide-area network (WAN).

Although expensive in absolute numbers and being a scarce resource, WAN serves a similar role in connecting data centers as power grid serves in connecting electricity consumers, and is better in two major ways: 1) it's easier to add more capacity in WAN than in power grid due to the presence of dark fibers; and 2) it's way more energy efficient to transmit data in these fiber-optics-based networks than transmit electricity over long-distance voltage lines, which can have high transmission losses.

Part of our goal in this project is to verify this high-level intuition and to accurately quantify the carbon cost of workload migration over WAN.

4.2.2 Background and carbon modeling

We now move onto the key metrics to quantify the carbon costs as well as our carbon cost model.

Carbon intensity of power grids and ISOs

Local power grids are run by Independent System Operators (ISOs) which are responsible for generating and supplying power to the grid. This energy has a *carbon intensity*, which is the grams of carbon dioxide generated per kilowatt-hour electricity (gCO2/kWh) [1]. We take our ISO carbon intensity numbers from Electricity Maps, a publicly available data provider [70]. We assume that datacenter power will have the same carbon intensity as the grid where it is physically located. This may not be completely accurate, as datacenter operators might have special agreements with ISOs which are not publicly available. However, this assumption is

[1]Carbon intensity is actually measured in gCO2**e**/kWh where the e stands for *equivalent*. This value fluctuates over time due to changes in the underlying fuel mix, but is often reported hourly. This is not a direct measurement of carbon dioxide emissions, but rather a metric that normalizes different greenhouse gases by the amount of warming they contribute. We use gCO2/kWh in this chapter for simplicity.

common across the community [111, 83, 97], and our technique does not rely on a specific data source, so the carbon intensity numbers can be updated easily.

CIDT: Carbon intensity of data transfer

Similar to how we measure the carbon emissions per kilowatt-hour electricity in carbon intensity, Tabaeiaghdaei et al. [97] proposed the concept of Carbon Intensity of Data Transfer (CIDT), which measure the per-gigabit carbon emissions of moving data across a network path [97]. This represents the total carbon emission per unit of data transferred across all the network devices along the path, and is usually measured in gCO2/Gb(it). We adopt the same concept in our analysis, and discuss the detailed calculation in Section 4.5.2. Note that CIDT is route (location) and time dependent.

Carbon emissions of workload migration

In this work, we consider both *compute* and *transfer* carbon costs for geographic workload migration. We focus on workload migration across large geographic regions, particularly between cloud provider's worldwide locations.

The *compute carbon cost* is the emissions generated during the compute phase of a workload. This is the power consumption of a computer "billed to" the workload multiplied by the carbon intensity of the energy where that computer is. By default, we use 5 W as an estimate for the power of a vCPU, based on recent Intel Xeon CPUs. The *transfer carbon cost* of migration we further break down into *network* transfer costs and *endpoint* transfer costs. Network transfer costs include all network devices enabling the data transfer, and endpoint transfer costs include the sending/receiving servers at source and destination region. In both cases, we proportionally "bill" the energy cost based on the usage, which is CPU utilization for compute and bandwidth for network. We use this simple model of energy billing for our analysis, and discuss the limitation in Section 4.7.

Network Transfer Costs

For network carbon cost, we adopted the same power modeling as [97], where routers consume most of the energy along the network path. They have shown that due to the high energy/bit cost of Internet core routers, higher Power Usage Efficiency (PUE) and redundancy, wide-area network carbon cost can be several orders of magnitude higher than intra-datacenter networks.

In this work, we further improve the accuracy and details of network paths, by combining their traceroute data with physical Internet infrastructure dataset. This allows us to reveal the hidden hops in traceroute measurement, and more accurately account for the energy cost of fiber-attached devices based on fiber map.

We also note that the network device powers in [97] are now more than ten years old. Since we don't know the exact equipment used in today's WAN networks, and to make a fair comparison, we used the same power numbers, but we discussed the potential improvement based on new devices today in Section 4.4.3.

Endpoint Transfer Costs

We model the endpoint transfer cost as the sum of power used on each end host during the transfer. We estimate 20% of a machine's power at each end as the endhost sending/receiving power, based on the networking cost model in modern data centers [17], and we multiply that by the carbon intensity to get the carbon emission values.

4.2.3 Motivating example

Now that we've described our carbon cost model, we will describe a concrete example showing the potential high impact of the network in carbon-aware workload migration.

We broadly categorize datacenter workloads into "compute-heavy" and "network-heavy". We first consider a network-heavy workload. Our hypothetical workload uses 10 CPU core-hours and has a total of 500 GB of input and output data. This workload is launched in Virginia, US, at

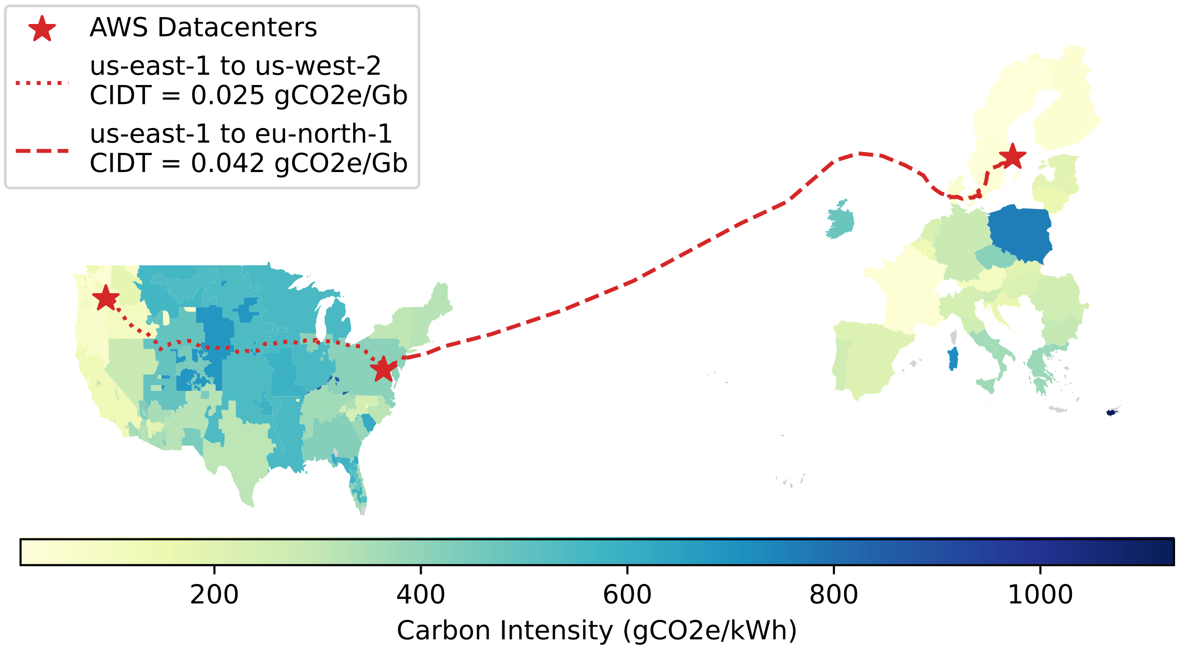


Figure 4.1. The carbon intensity of different regions in the northern hemisphere on June 1st, 2023. The routes between AWS datacenters `us-west-2`, `us-east-1`, and `eu-north-1` are shown with their CIDTs.

the AWS `us-east-1` datacenter at noon eastern time on June 1st, 2023.

What are the carbon emissions of this workload? Figure 4.1 shows the carbon intensity (in gCO2/kWh) of energy grid sources in the US at the time the workload is launched. AWS `us-east-1` is serviced by PJM Interconnection, which uses a combination of nuclear, coal, and natural gas. At noon on June 1st, the carbon intensity of the energy for `us-east-1` is around 400 gCO2/kWh. This means that a 1 hour job which consumes 1 kilowatt of power would emit 400 gCO2, the equivalent of burning around 0.2 liters of gasoline [45]. Our workload which consumes 10 CPU core-hours would consume 0.05 kWh energy and emit around 20 gCO2, assuming that each core consumes 5 Watts of power [55].

The carbon intensity of energy in the US is not uniform, and at the same time, the carbon intensity of the AWS `us-west-2` datacenter was lower. The AWS `us-west-2` datacenter is in Washington, which is served by a combination of natural gas and hydropower. At the time our job launched, the carbon intensity of this energy was around 100 gCO2/kWh, which would lead to a reduction in carbon emissions of around 75%. If we wanted to minimize carbon emissions, it may be better to run our job in Washington, assuming there is no latency requirement.

However, this workload has input data and produces output data. To run in `us-west-2`, we would also need to transfer this data, which itself produces carbon emissions. Some of the devices on this path are served by ISOs that use relatively higher carbon energy. In this example, the CIDT of this path is 0.025 gCO2/Gb. This means that sending the data required for our job produces around 100 gCO2. So even though we would save 75% of the compute carbon emissions, our overall emissions would be over 5$\times$ as high, because of the high cost of transfer.

Compute Heavy Example:

The previous workload is very network heavy, so we also consider a more compute-heavy workload. This workload has 100 CPU core-hours and 500 GB of input and output data. The AWS `eu-north-1` datacenter in Sweden has the lowest carbon intensity anywhere in the world at this time. If the network is ignored, then sending our workload to `eu-north-1` will lead to the overall lowest carbon emissions at around 10 gCO2. However, to migrate our job to `eu-north-1`, we would incur around 170 gCO2 of migration costs, for a total of around 180 gCO2. In contrast to this, the `us-west-2` datacenter would release 25 gCO2 for compute, but only 100 gCO2 for migration. Even though the carbon intensity of the energy is $\sim 2\times$ as high in `us-west-2`, the overall carbon emissions are around 30% lower there.

These examples show that workload migration has the potential to reduce carbon emissions for certain types of applications, but the carbon emissions of the network cannot be ignored if we want to choose the optimal region.

4.3 Related work

A number of efforts have looked into selecting network paths based on carbon intensity data. Zilberman et al. [125] argue for a carbon-aware network approach to considering the combined carbon of both compute as well as wide-area network transfer costs. In this work, they outline a number of metrics that are important in making carbon-aware path selections. El-Zahr et al. [118] likewise consider the carbon cost of network paths, and propose a “Carbon-aware traffic engineering algorithm” called CATE that aims to minimize carbon emissions. As mentioned earlier in the chapter, Tabaeiaghdaei et al. [97] forecast carbon conditions along network paths and use the SCION path-aware architecture to choose paths that lower carbon emissions.

Since the availability of lower-carbon energy sources varies over time, a number of efforts have examined the ability to delay workloads with looser time requirements to times when lower carbon energy is available. Both Meta [52, 4] and Google [82, 83] explore delaying workloads, or shifting work up in time, to better match diurnal carbon cycles. Wiesner et al. [111] focus on temporal workload shifting and identify delay-tolerant workloads that are suitable for time shifting. Hanafy et al. [50] exploit the elasticity in batch workloads and propose “carbon scaling” using existing cloud autoscaling mechanisms.

Of particular relevance to our work are approaches that migrate data and compute geographically to better match low-carbon energy sources (or alternatively, load balance requests to replicas in regions with low-carbon energy). Agarwal et al. [6] propose moving computation instead of energy in response to changes in grid condition, in a group of sites that have complimentary power availability and enough WAN bandwidth. Zheng et al. [122] show the potential carbon savings of migrating workloads between data centers in locations with different grid carbon intensities.

From grid perspective, Liu et al. [67] explore how factoring in the carbon intensity of grid conditions on load balancing decisions can reduce the overall carbon of the composed system, and Lin et al. [63] further emphasize the necessity for data center operators to coordinate with the grid to achieve successful carbon savings rather than harming the grid.

At platform level, Zhang et al. [121] propose geographical load balancing algorithms that dynamically route requests to data centers with lots of renewable energy, while staying within quality of service requirements, renewable availability, datacenter capacity and WAN budget constraints. Souza et al. [95] advocate for a new API to allow application access of carbon metrics and control of their power usage in response. Shen et al. [91] propose Supercloud, which is a live virtual machine inter-cloud migration system that could incorporate carbon intensity as a metric for selecting where to migrate code and data.

4.4 Network carbon costs and routes

To calculate the carbon costs of data transfer across the wide-area network, we need to know the route that data takes, along with the physical devices along that route, and the carbon intensity of the ISO powering those devices. Previous solutions [97] have used traceroute data to infer routes, devices, and the location of those devices. In this section, we discuss this approach along with its limitations, and describe how we use additional data sources to improve our network carbon costs.

4.4.1 Straw-man solution: `traceroute`

Prior work [97] calculates the carbon cost of default BGP path (for a given source and destination) by 1) running shortest path algorithm on CAIDA ITDK dataset [101], which is a graph of Internet routers and their connectivity, 2) translate each router IP into geolocation using the builtin IP-to-geo table, a stripped-down version of MaxMind [71], and 3) Convert geolocation to ISOs. Total network carbon cost of the route, or CIDT, can be calculated by aggregating power usage per device multiplied by the carbon intensity at each device.

We start with the same approach to find the shortest routes across all AWS and Google Cloud regions (19 in AWS and 29 in Google Cloud). We use the public IP prefixes in each AWS and Google Cloud region, and match them with the router IPs in the ITDK dataset.

However, we found that this approach, although easy to analyze and reproduce, can suffer from several limitations: 1) traceroute cannot reveal all physical hops due to the use of MPLS [107], especially in private links used by cloud providers [115, 108]; 2) IP-to-geo datasets often lack in coverage and accuracy, resulting in low fidelity routes, and 3) without the knowledge of fiber paths, auxiliary network devices are placed like fiber amplifiers uniformly on a straight line between router hops. Similarly, in case traceroute fails to find a path, routers are estimated based on the distance from source to destination, which can be inaccurate.

Limitation #1: Hidden hops

Because ITDK is based on a collection of traceroutes from public probes, it only contains routes visible on the IP layer over the public Internet. However, cloud providers often utilize private networks. Because of this and the presence of MPLS tunnels, traceroute cannot accurately reveal the physical presence of routers on a wide area route [115]. From a network carbon cost accounting perspective, this can significantly skew the result, as routers consume most power of a network path [97].

We attempted to run traceroute from within the cloud and to reveal the MPLS tunnels using [108], but all intra-cloud traceroutes in Google Cloud show no intermediate hops between source and destination, and we revealed less than 1% of the MPLS tunnels across all routes between AWS and Google Cloud regions.

Limitation #2: Inaccurate geolocation

Another issue with ITDK and traceroute is the coverage and inaccuracy of IP geolocation. This is inherent due to the nature of the data collection and is seen in all data sources.

In our analysis, we find that routes using ITDK's builtin geolocation data (`itdk.node.geo`) can often lead to unexpected locations and cause longer routes. Upon investigation and checking the source MaxMind dataset [71], we figure out that these are the default values when the accuracy is low (Kansas for US, Stockholm for EU), and ITDK's geolocation data omits the accuracy radius. To get around this issue, we directly query MaxMind and remove routes in which any router's geolocation accuracy is greater than 100 km.

The resulting set of routes still have a lot of "noise". We further remove routes that do not make sense by: 1) filtering out the routes that do not start/end with the correct regions, and 2) removing routes greater than 2$\times$ great-circle distance. We determine the correct start/end region using the ground truth city location of AWS and Google Cloud [85, 86]. Finally, we break ties by choosing the most popular route that is at least $1.2\times$ the minimum distance.

Limitation #3: No physical path

A third disadvantage of this approach is that traceroute provides no physical path information. Not only are we ignoring the hidden routers inside private links and MPLS tunnels, we are also underestimating fiber lengths and potentially mapping fiber-attached amplifiers to the wrong ISOs. Cross-continent submarine cables also consume more power due to higher power loss, as they can only be powered from either end. This means that we are not able to accurately infer the location of auxiliary devices along the path.

4.4.2 Inferring physical routes

From the perspective of carbon analysis, there is a pressing need for the physical location of network devices. This is because carbon metrics are based on ISOs, which map to regions of the physical world. In contrast, traceroute measures are at IP levels, and do not accurately describe the physical presence due to the aforementioned limitations.

Since we are focusing on routes over the wide area, we turn to existing Internet measurement studies to help us discover the physical location of routers on WAN links. In particular, we use the dataset from iGDB [11], a recent work on Internet route mapping using combined physical and logical measurement data, to reconstruct the routes between cloud region pairs at city-level as a graph. We use cities of physical entities as nodes and the fiber paths among these entities as edges. For nodes, the entities we consider include public peering locations (e.g. from PeeringDB [78]), and optionally the Point of Presences (PoPs) of AWS and Google Cloud, to deduce this information in a reasonably accurate fashion. Our hypothesis is that even though AWS or Google Cloud can own their own fiber, the underlying physical fiber conduits are often shared, and they likely want to exchange traffic with other Internet providers at major peering cities.

We do, however, have to heavily reprocess iGDB's dataset to connect both land and submarine fiber cables, as 1) the original fiber data for these two comes from different sources, and we need to identify land and submarine differently for power attribution, so they cannot be directly connected to build a single graph, 2) many cloud regions are not near a major peering city, so we manually "tap" them into nearest fibers, and 3) we also optionally add cloud provider's own Point of Presence (PoP) locations along the shortest path.

This allows us to calculate the shortest paths between any city pairs, as well as to reconstruct the physical fiber path between each hop. We refer to routes calculated using this method as *inferred physical path*, or *inferred path* for short.

4.4.3 Combine `traceroute` and inferred paths

Now that we have a way to infer physical paths between two cities, we propose two strategies for using it.

One solution is that we directly infer paths based on the endpoints, i.e. source and destination region. Here, we ignore the traceroute data entirely, and purely infer the path based on the physical Internet infrastructure. We call this method **Inferred based on endpoints**.

Another solution is to use the hops in traceroute as waypoints. In this case, we filter out the inaccurate locations as described in Section 4.4.1, and consider the high-confidence hops as waypoints. We can then iteratively infer the paths between each pair of consecutive waypoints based on the approach outlined by Anderson et al. [11]. In this case, we consider both traceroute data and physical Internet infrastructure. We call this method **Inferred based on waypoints**.

Comparing route accounting methods

We list the information available in each of the three route accounting methods in Table 4.1. Traceroute refers to the original route accounting methods described in [97] for comparison purposes. As we can see, traceroute and inferred path based on endpoints each

[2]Note that the original traceroute data in [97] (CAIDA ITDK [101]) does not include IP-to-geo accuracy data.

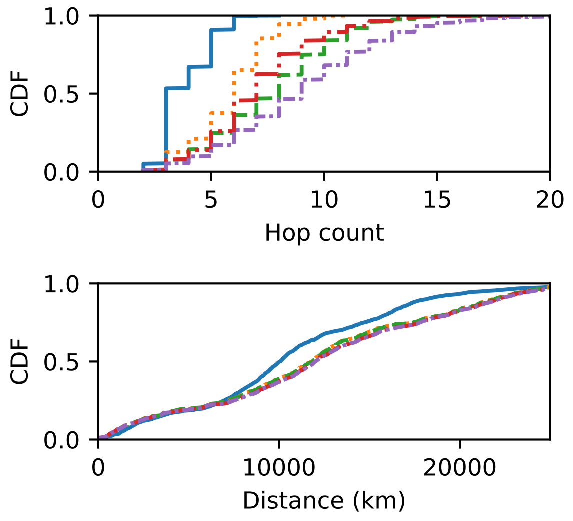
Table 4.1. Comparison of different route accounting methods.

Information	traceroute	Inferred (endpoint)	Inferred (waypoint)
Endpoints	x	x	x
traceroute	x		x
IP-to-geo	x [2]		x
Infrastructure		x	x
Fiber paths		x	x
PoPs		x	x

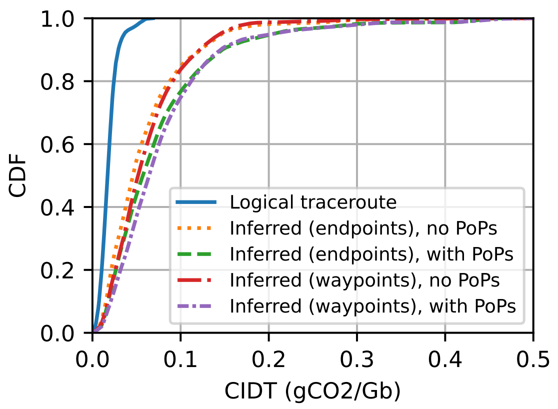
covers a different set of information, all of which can be useful. For example, traceroute can indicate intermediate hops that are not on the shortest paths, possibly due to the WAN routing policy in use. On the other hand, Internet infrastructure and Point of Presence (PoP) data enable us to deduce hidden hops that traceroute cannot. Thus, we believe the combination of these two datasets, i.e. **inferred path based on waypoints is the most accurate estimate** of wide area routes for carbon accounting purposes.

We evaluated all three methods, and in the two cases of inferred route accounting methods, whether or not we include the PoPs of AWS and Google Cloud, for a total of five methods. We performed this evaluation across all pairs of the 48 AWS and Google Cloud regions, and plotted the distribution of hop count and distance in Figure 4.2a and the distribution of network carbon cost, or CIDT, in Figure 4.2b.

We can see that inferred paths generally have more hops and higher CIDT than traceroute, which is consistent with the observation of hidden hops. They also have slightly longer distances because they use accurate fiber path map instead of direct line distance. Inferred path based on waypoints have slightly higher hop count and CIDT because we also consider the high-confidence waypoints from traceroute. Lastly, inferred paths with PoPs have higher hop count than those without PoPs. This is because cloud providers PoPs do not show up in public peering location data. We believe adding these locations help build a more accurate view of the physical hops, as AWS traffic are highly likely to stop at an AWS Point of Presence. **Thus for our evaluation, we choose inferred path based on waypoints with PoPs.**



(a) Hop count and distance



(b) Network Transfer Carbon Cost in 2023

Figure 4.2. Comparison of all route accounting methods.

Comparing the CIDT distribution with the default BGP route in Figure 4 of [97]. Tabaeiaghdaei et al. measures a median of about 0.035 gCO2/Gb, whereas we found that across all AWS and Google Cloud region pairs, the median is about 0.017 gCO2/Gb. Our result is lower but the difference is likely due to relative closer distance between the region pairs we consider vs the entire Internet analyzed in [97]. Comparing the traceroute method against the more accurate inferred path based on waypoints method, we observe about $2-3\times$ difference in the median case, and much higher difference at higher percentiles.

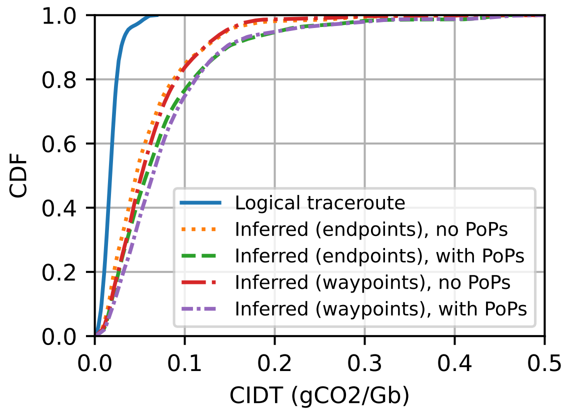
Improvement in power consumption of network equipment: 2010-era vs now

We would like to note that the CIDT results presented above are based on decade-old power consumption numbers for network equipment. Thus, we also present the CIDT results using more current power consumption numbers.

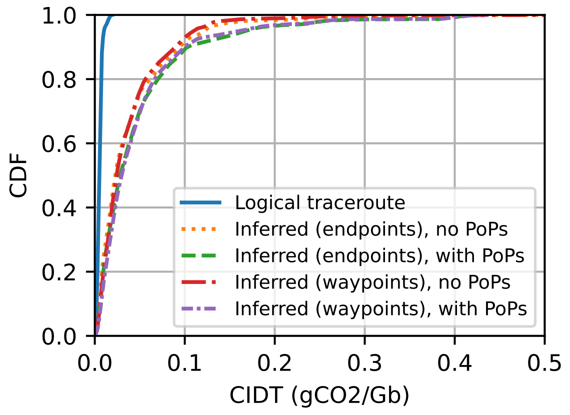
In Table 4.2, we first compare the 10-year old numbers from Table 2 of prior work [97], i.e. [106], and newer numbers, where the main difference are the improvement of core routers from 10 W/Gbps to $1-2$ W/Gbps [105] and that of transceivers from 1.5 W/Gbps to 0.09 W/Gbps [2]. We then show the improved CIDTs in Figure 4.3. But to make consistent compar-

Table 4.2. Comparison of the per-Gbps power consumption of 2010-era [106] and current network devices.

Device	Per-Gbps power consumption (W/Gbps)	
	2012	2024
Core router	10	2
WDM switch (OXC)	0.05	0.05
Transceivers	1.5	0.09
Amplifier	0.03	0.03
Regenerator	3	3



(a) Using 2010-era power numbers



(b) Using 2020-era improved power numbers

Figure 4.3. Comparison of CIDT using old and new power numbers (Table 4.2)

isons, we use the same 2010-era power numbers for the rest of this study.

4.4.4 Route dataset

Based on these route accounting methods, we have collected and curated a dataset of routes between all pairs among the 48 AWS and Google Cloud regions. This includes the number of hops, the type of the device at each hop, and the distance and physical fiber path between each pair of consecutive hops. We plan to open source the full result of our route dataset, but due to the large volume of data, here we show the schema and a preview of the dataset.

Between each pair, defined as a 4-tuple (`src_cloud`, `src_region`, `dst_cloud`, `dst_region`), and for each route accounting method (`source`), we capture:

- `hop_count`: # of hops/routers
- `distance_km`: total distance in km (great circle distance for `traceroute` method)
- `routers_latlon`: delimited list of routers' coordinates, in (lat, lon) format.
- `fiber_wkt_paths`*: standard WKT path string, one segment per hop. Note that WKT uses (lon, lat) coordinates.
- `fiber_types`*: delimited list of the fiber types for each segment of `fiber_wkt_paths`; either `land` or `submarine`.

Note: items marked with * are only available for inferred path methods, as they come from the physical fiber map.

Preview of the dataset:

- `src_cloud`: aws
- `src_region`: us-east-1
- `dst_cloud`: aws
- `dst_region`: eu-north-1

- `hop_count`: 9
- `distance_km`: 7889.28
- `routers_latlon`: (39.0469, -77.4903) | ...
- `fiber_wkt_paths`: MULTILINESTRING ((-77.4903 39.0469, -77.01376 38.89731), ...)
- `fiber_types`: land | ... | submarine | ... | land
- `source`: Inferred (waypoints), with PoPs

4.4.5 Carbon data availability

As part of calculating the CIDT across all these region pairs, we discovered that the more detailed route accounting method, i.e. inferred path based on waypoints, generates a larger set of ISO due to the large amount of low-power auxiliary devices along the fiber. This creates a problem where some of the network devices are in a lesser known ISO with poor carbon data coverage. [3] We can discard the routes where carbon data is not available, but in the worst case, we have only half of the routes with known carbon data. Upon a deeper look, we found that in many cases, the majority of the network devices (by power) have known carbon data, which we extrapolate to the whole path.

Estimating CIDT with partial carbon data

For each route, we first calculate the ratio of power with carbon data over the total power across all network devices. If this ratio is too low, then we ignore the route as any estimation will likely incur too much error and skew our result. We considered both a 50% and 75% cutoff threshold for this ratio. We then compare two heuristics:

[3]In particular, this is because electricity map only publishes full year data for a smaller set of 130 regions that are popular, out of the full 400 regions around the world.

- Route Average: we calculate the route average CIDT for devices with carbon data and if that's less than 100%, we scale this value to 100%.
- Nearest Neighbor: for each device without carbon data, we use the carbon intensity from the nearest neighbor with carbon data.

Table 4.3. Estimated network carbon cost accuracy.

Heuristic	Coverage	Error Percentage (%)		
		Mean	90%-tile	99%-tile
50% Cutoff				
Route Average	95.6%	5.35	19.76	56.77
Nearest Neighbor	95.6%	5.22	17.09	58.87
75% Cutoff				
Route Average	84.2%	2.56	9.58	28.59
Nearest Neighbor	84.2%	2.60	8.86	31.13

We use the most recent 60 days where we have carbon data for all the ISOs to calculate the ground truth CIDT, and hide the carbon data for the ISOs which we don't have full year data to calculate the error percentage. We summarized the coverage in # of region pairs and estimation error in Table 4.3. We found the estimation error is very similar between the two heuristics, in both 50% and 75% cutoff. We chose the route average method, because our analysis found that the nearest neighbor with carbon data can be hundreds to thousands of kilometers away (only 70.8% coverage at $\leq$ 500 km limit).

We do find that by limiting the cutoff of known-carbon power ratio to $\geq$ 75%, we can dramatically reduce the estimation error, while still maintaining 84.2% coverage (baseline without any estimation is 58.4%).

4.5 Joint optimization with compute and transfer carbon cost

Now, we will formally describe the mathematical optimization problem based on our carbon cost model (Section 4.2.2).

At a high level, our goal is to find the lowest total carbon emission of a job for each region, which consists of compute, and in the case of migration, transfer carbon emission. We can then find the optimal region with lowest combined compute and transfer carbon emission. These values depend on several factors, including the job characteristics and time-varying carbon intensity and CIDTs.

4.5.1 Job characteristics

We assume that a job has a few parameters that we know at scheduling time: `runtime`, `core_count`, `start_time`, `deadline`, `input_size`, `output_size`, `origin`. These parameters are often either provided by users to job schedulers, or can be profiled at the runtime, so we assume they are known.

They set the constraints of this optimization problem and determine the energy usage and the transfer time.

For our analysis, we assume a fixed network transfer bandwidth at $B = 1$ Gbps, which has been demonstrated to be viable in today's commercial cloud at low cost [56]. Thus the transfer durations can be calculated as $D_{Input} = \texttt{input\_size}/B$ and $D_{Output} = \texttt{output\_size}/B$ (or 0 for original location). Similarly, we denote $D_{Compute} = \texttt{runtime}$ as the duration of compute.

4.5.2 Carbon intensity and CIDT time series

We also need to know the time-varying *Carbon Intensity of Electricity* (CIE) and *Carbon Intensity of Data Transfer* (CIDT), in order to translate energy usage to carbon emissions, for compute and transfer respectively.

Note that both are time-dependent, and in addition, CIE is region-dependent, and CIDT is route-dependent. In our analysis, since we assume a fixed route per source/destination region pair, we can say CIDT is region-pair-dependent.

We denote $CIE_r(t)$ as the carbon intensity at region r at time t, usually measured in gCO2/kWh, and this is provided by ISOs or commercial providers like electricity map [70].

We denote $CIDT_{src \to dst}(t)$ as the total CIDT (in gCO2/Gb) from *src* to *dst* across all network devices. We calculate this by aggregating the per-hop CIDT across all routers and auxiliary network devices $\mathbb{D}$, which we generated similarly as described in [97], but using our inferred paths:

$$CIDT_{src \to dst}(t) = \sum_{D \in \mathbb{D}} \frac{P_{max,D}}{C_{max,D}} * CIE_D(t)$$

Here, $P_{max,D}$ is the maximum power of a network device D and $C_{max,D}$ is its maximum capacity. We use a similar proportional energy accounting method on the network devices, although other alternative options exist but this is outside of the scope of this work.

4.5.3 Optimization variables

In time-shifting, there is one scheduling decision, that is when to start the job. Let's denote that as $t_{Compute}$.

In space-shifting, because there's also a time-varying cost of transfer based on CIDT, we also need to optimize when to start input and output transfers, which we denote as t_{Input} and t_{Output} respectively.

Note that input transfer, compute and output transfer must happen in order and not overlap. This sets the constraints:

$$\begin{aligned}
\texttt{start\_time} &\leq t_{Input} \\
t_{Input} + D_{Input} &\leq t_{Compute} \\
t_{Compute} + D_{Compute} &\leq t_{Output} \\
t_{Output} + D_{Output} &\leq \texttt{deadline}
\end{aligned}$$

Now, given the source region $r_0 = \texttt{origin}$ and a list of alternative regions $r_i \in \mathbb{R}$, we will define the carbon emission values next.

4.5.4 Compute carbon emission

For any region $r \in \{r_0\} \cup \mathbb{R}$, we denote the compute carbon emission of the job (in gCO2) as:

$$CE_{Compute,r} = \int_{t_C}^{t_C+D_C} CIE_r(t) * P_C * PUE_{DC} \, \mathrm{d}t$$

Here, compute power $P_C = \texttt{core\_count} * \texttt{watts/core}$, and t_C and D_C are the start time and duration of compute. We use 5 W/core as the default based on recent Xeon CPUs, but this is customizable.

4.5.5 Data transfer carbon emission

For the original location r_0, the transfer cost $CE_{Transfer,r_0}$ is zero, since we don't need to migrate the data.

For all other alternative regions $r \in \mathbb{R}$, we compute the total transfer cost as the sum of network and endpoint:

$$CE_{Transfer,r} = CE_{Network,r} + CE_{Endpoint,r}$$

We compute the network carbon cost $CE_{Network,r}$ using CIDT of the routes:

$$\begin{aligned} CE_{Network,r} = & \int_{t_I}^{t_I+D_I} CIDT_{r_0 \to r}(t) * B \, \mathrm{d}t \\ & + \int_{t_O}^{t_O+D_O} CIDT_{r \to r_0}(t) * B \, \mathrm{d}t \end{aligned}$$

Here, t_I/t_O, D_I/D_O are the start time and duration of input/output, and B is the bandwidth.

We compute the endpoint carbon emission by integrating the carbon intensity with the transfer power, and we do this for both sending and receiving data, at source and destination respectively. We assume transfer power $P_{Transfer}$ to be 20% of a server based on [17].

$$
\begin{aligned}
CE_{Endpoint,r} = & \int_{t_I}^{t_I+D_I} (CIE_{r_0}(t) + CIE_r(t)) * P_{Transfer}\,\mathrm{d}t \\
& + \int_{t_O}^{t_O+D_O} (CIE_r(t) + CIE_{r_0}(t)) * P_{Transfer}\,\mathrm{d}t
\end{aligned}
$$

4.5.6 Optimization goal

Given this problem formulation and constraint, the goal is to schedule the transfers and compute to get the lowest carbon emission on a per region basis.

$$CE_r = \min_{t_I, t_C, t_O} CE_{Compute,r} + CE_{Transfer,r}$$

And the best region has lowest total carbon emission:

$$\texttt{best\_region} = argmin_{r \in \{r_0\} \cup \mathbb{R}} CE_r$$

4.5.7 Linear time optimization algorithm

Because we want to optimize for the total amount of carbon emissions from both compute and transfers, we need to simultaneously optimize for three time variables instead of one: namely t_I: when to start the input transfer, t_C: when to start the compute, and t_O: when to start the output transfer.

Although it is easy to use a simple sliding window algorithm when not considering the two data transfers, it is not a straightforward task to optimize for three time variables in an efficient manner. An naive algorithm will have to scan for all possible combinations of three timing variables, which runs in $O(n^3)$ time. This does not scale with our real-time optimization requirement across multiple regions.

However, we note that carbon intensity and CIDT values are only updated hourly. Based on this insight, we developed a dynamic-programming-based linear time algorithm by pre-computing the individual carbon emissions at any given starting time, for all three components, and then linearly scan through possible start times of the middle component, i.e. start of the compute job. Mathematically, we describe this linear time algorithm in Algorithm 1, and include the two auxiliary functions `CalculateIntegral` and `GetOptimalPoints` in Algorithm 2.

This linear time algorithm enables us to perform efficient analysis across a whole year at 4-hour granularity on a multi-core machine in minutes.

Algorithm 1. An $O(n)$ algorithm for optimizing compute + transfer carbon emissions.

```
Input: $f_1, f_2, f_3 : \mathbb{T} \to \mathbb{R}^+$: Step functions of carbon emission rates of input transfer, compute
    and output transfer
Input: $T_0$: The initial start time
Input: $T_4$: The maximum allowed end time
Input: $D_1, D_2, D_3$: The duration of input transfer, compute and output transfer
Output: $t_1, t_2, t_3$: Start times of input transfer, compute and output data transfer
        Constraint: $T_0 \le t_1 < t_1 + D_1 \le t_2 < t_2 + D_2 \le t_3 < t_3 + D_3 \le T_4$
        Objective: $argmin_{t_1,t_2,t_3} \int_{t_1}^{t_1+D_1} f_1(t)\,\mathrm{d}t + \int_{t_2}^{t_2+D_2} f_2(t)\,\mathrm{d}t + \int_{t_3}^{t_3+D_3} f_3(t)\,\mathrm{d}t$
1: for $i \in \{1,2,3\}$ do
2:     $f_i^I(t) \leftarrow$ CalculateIntegral($f_i, D_i, T_0 + \sum_{i'=0}^{i-1} D_{i'}, T_4 - \sum_{i'=i}^{3} D_{i'}$)
3:     if $i = 1$ then
4:         $OP_i \leftarrow$ GetOptimalPoints($f_i^I, T_0 + \sum_{i'=0}^{i-1} D_{i'}, T_4 - \sum_{i'=i}^{3} D_{i'}$, false)
5:     else if $i = 3$ then
6:         $OP_i \leftarrow$ GetOptimalPoints($f_i^I, T_0 + \sum_{i'=0}^{i-1} D_{i'}, T_4 - \sum_{i'=i}^{3} D_{i'}$, true)
7: $min\_integral_{total} \leftarrow$ infinity
8: $T_{optimal} \leftarrow \{NaN, NaN, NaN\}$
9: for $t_2 \in [T_0 + D_1, T_4 - D_2 - D_3]$ do
10:                                                  ▷ Calculate minimum $integral_1$
11:     $t_1\_max \leftarrow t_2 - D_1$
12:     $op_1 \leftarrow \max(\{op \in OP_1 | op \le t_1\_max\})$
13:     $optimal\_t_1 \leftarrow argmin_{t \in \{op_1, t_1\_max\}} f_1^I(t)$
14:     $min\_integral_1 \leftarrow f_1^I(optimal\_t_1)$
                                                     ▷ Calculate minimum $integral_3$
15:     $t_3\_min \leftarrow t_2 + D_2$
16:     $op_3 \leftarrow \min(\{op \in OP_3 | op \ge t_3\_min\})$
17:     $optimal\_t_3 \leftarrow argmin_{t \in \{op_3, t_3\_min\}} f_3^I(t)$
18:     $min\_integral_3 \leftarrow f_3^I(optimal\_t_3)$
                                                     ▷ Compare total integral
19:     $integral_{total} \leftarrow min\_integral_1 + f_2^I(t_2) + min\_integral_3$
20:     if $integral_{total} < min\_integral_{total}$ then
21:         $min\_integral_{total} \leftarrow integral_{total}$
22:         $T_{optimal} \leftarrow \{optimal\_t_1, t_2, optimal\_t_3\}$
23: return $T_{optimal}$
```

Algorithm 2. Auxiliary functions for the optimization function in Algorithm 1.

```
24: procedure CALCULATEINTEGRAL($f, D, T_{min}, T_{max}$)
    ▷ Calculate the integral of step function f over a fixed-duration moving window, to produce
    the total carbon emission.
25:     $f^I(t) \leftarrow \int_t^{t+D} f(t)\,\mathrm{d}t, \forall t \in [T_{min}, T_{max}]$        ▷ Integrate $f$ over duration $D$
26:     return $f^I(t)$
27: procedure GETOPTIMALPOINTS($f^I, T_{min}, T_{max}, reverse$)
    ▷ Get the series of times at turning points of the curve $f^I$ with non-increasing values.
28:     $OP \leftarrow \{\}$
29:     $last\_value \leftarrow infinity$
30:     if $reverse$ then $interval \leftarrow [-T_{max}, -T_{min}]$ else $interval \leftarrow [T_{min}, T_{max}]$
31:     for $t' \in interval$ do
32:         $t \leftarrow |t'|$
33:         if $f^{I\prime}(t-\delta) = f^{I\prime}(t+\delta)$ then        ▷ Not a turning point, ignoring
34:             continue
35:         if $f^I(t) \leq last\_value$ then
36:             $last\_value \leftarrow f^I(t)$
37:             $OP \leftarrow OP | \{t\}$
38:     return $OP$
```

4.6 Evaluation

In this section, we evaluate the impact of geographical workload shifting using three classes of workloads that vary in their ratio of compute to networking, which we call *low*, *medium*, and *high* data usage workloads. Geographic migration of work is suitable for workloads with more relaxed deadlines and response time requirements. As such, we focus on batch workloads with completion deadlines in the range of minutes to hours, with a maximum completion deadline of 24 hours (similar to Radovanovic et al. [82]). We leave jobs with longer deadlines for future work.

4.6.1 Workloads and impact on space-shifting

In this section, we describe our selection of data center workloads and the resulting impact on the effectiveness of space-shifting solutions.

Workload descriptions

We selected representative cloud workloads that span the range of low, medium, and high data requirements. For this study, we chose (1) video resizing, (2) video processing, and (3) source code compilation, since they have relaxed completion deadline requirements and benefit from the parallelism and capability of public cloud platforms. For each workload, we profiled their compute requirements using Intel RAPL on a server with a Xeon Gold 6138 CPU (with hyper-threading on) and 256 GB of memory, running Ubuntu 20.04 LTS. We chose this configuration since it has a similar watts-per-core ratio as common virtual CPUs (vCPUs) used in AWS and Google Cloud's cloud offering. We then chose the workloads with different compute heavinesses, as shown in Table 4.4. We also include a workload with a low data usage but long runtime, to study the effect of runtime.

We restrict our attention to these four jobs since they span a diverse range in their data size to compute energy usage ratio, which ultimately dictates the relative cost of migration. We see in Section 4.6.1 that the average migration overhead spans from less than 10% (in the case of a compilation workload) to 100% (for video resize). Beyond that point, the benefits of geographic migration diminish. We expect similar conclusions to be drawn for other practical applications when their data size to compute energy usage ratio and runtime are known.

Table 4.4. Workload definitions, with the last column "Ratio" representing the calculated *data size to compute energy* ratio, in Gb/kWh. We use both short and long versions of code compilation to study the effect of different runtime.

Workloads	Cores	Runtime (h)	Input (GB)	Output (GB)	Ratio (Gb/kWh)
Video resize	4	2.2	138.0	7.5	334.6
Video effect	8	2.1	30.0	3.1	38.8
Compile (short)	40	2.2	24.0	3.9	6.4
Compile (long)	40	11.9	132.0	21.5	6.4

Workload characteristic and impact

In this section we evaluate what kind of jobs benefit most from migration, based on their compute and data requirements. As we have shown in Section 4.5, the total carbon cost of a job is the sum of compute and transfer. The former is proportional to the energy usage of a job, measured in kWh, and the latter is proportional to the amount of data that needs to be transferred between regions, measured in Gb.

Thus, the net saving percentage by moving a job from r_0 to an alternative location r is represented by:

$$\begin{aligned}
\text{Net saving }(\%)_r &= \frac{CE_{r_0} - CE_r}{CE_{r_0}} \\
&= \frac{CE_{Compute,r_0} - (CE_{Compute,r} + CE_{Transfer,r})}{CE_{Compute,r_0}} \\
&= (1 - \frac{CE_{Compute,r}}{CE_{Compute,r_0}}) - \frac{CE_{Transfer,r}}{CE_{Compute,r_0}} \\
&= \underbrace{(1 - \frac{\int CIE_r \,\mathrm{d}t}{\int CIE_{r_0} \,\mathrm{d}t})}_{\text{Compute Savings }(\%)} - \underbrace{\frac{CE_{Transfer,r}}{CE_{Compute,r_0}}}_{\text{Migration Overhead }(\%)}
\end{aligned}$$

The first part is compute savings, which is purely dependent on the carbon intensity differences, and has been analyzed in many time shifting solutions. However, migration overhead has often been ignored or simplified.

If we break this into network and endpoint, we find that:

$$
\begin{aligned}
\text{Overhead } (\%) &= \frac{CE_{Transfer,r}}{CE_{Compute,r_0}} = \frac{CE_{Network,r} + CE_{Endpoint,r}}{CE_{Compute,r_0}} \\
&\approx \frac{D_T * \overline{CIDT} * B + (\overline{CIE_r} + \overline{CIE_{r_0}}) * D_T * P_T}{D_C * P_C * \overline{CIE_{r_0}}} \\
&= \frac{\texttt{data\_size}}{\texttt{compute\_energy}} \\
&\quad * \frac{(\overline{CIDT} + 1/B * (\overline{CIE_r} + \overline{CIE_{r_0}}) * P_T)}{\overline{CIE_{r_0}}}
\end{aligned}
$$

$D_T = D_I + D_O$ is the total data transfer time, $\texttt{data\_size} = \texttt{input\_size} + \texttt{output\_size} = D_T * B$ is the total data size of the job, and $\overline{CIDT}$ and $\overline{CIE}$ are the average values across the integral to simplify the equation.

Note that only the first term, namely the ratio of *data size* over *compute energy* depends on the job, whereas variables in the second term such as CIDT and CIE are job-agnostic. Thus, from the perspective of jobs, we can see that the lower the *data size / compute energy ratio* [4] is, the less migration overhead there is. That is to say, given a time, route and region(s), *the migration overhead of a job is proportional to the ratio of data size over the compute energy*. So the question we want to answer is, when is the ratio too high or too low?

To answer this, we take the average carbon intensity of 475 gCO2/kWh [54] and median, 25%-tile and 75%-tile CIDT across all region pairs (Figure 4.2b), and calculate the migration overhead percentage as we vary the data size to compute energy ratio. We plot the result in Figure 4.4, and overlay a few example jobs with varying ratios. To simplify the term, we will refer to low ratio jobs as *low data* jobs, as they consume relatively less data per unit of compute energy and thus incur lower relative migration carbon cost. On the other end, we call jobs with higher ratio *high data* jobs; these jobs generate more carbon emissions from migration. We can see that compilation, a low data job might incur $\leq 10\%$ migration overhead, whereas a medium

[4]The inverse of this concept, *compute-to-data-size ratio*, and preliminary analysis of migration overhead were previously published as a poster [48].

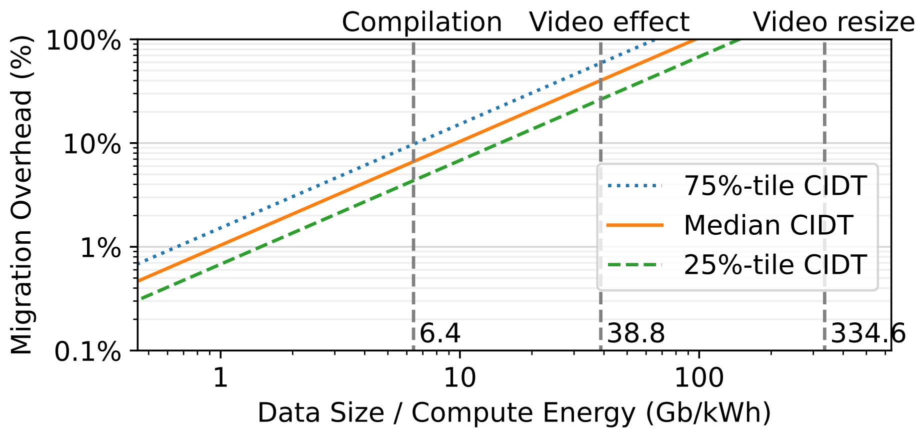


Figure 4.4. Migration overhead using different CIDT values.

data job like video effect can pay around 50% overhead, and a high data job like video resize might not make sense to migrate at all.

4.6.2 Time shifting vs space shifting

Now that we have selected a set of candidate jobs and understand the relationship between compute savings and migration overhead, we can start to evaluate the effectiveness of space-shifting.

To start with, we would like to understand whether space-shifting is more or less beneficial than existing time-shifting solutions [111, 83]. We picked two data centers Figure 4.5 shows the percent carbon emissions reduction for only time shifting, and for our approach which allows for time and space shifting. For the `us-west-1` datacenter and the `eu-central-1` datacenter, we simulated our four workloads with a 4 hour deadline and a 24 hour deadline. The workload request time was swept from 06/01/2023 to 06/29/2023 in 1 hour increments. The hourly reduction in carbon emissions were then averaged and plotted in Figure 4.5. Both of these regions have relatively high usage of solar energy, leading to large daily variation in carbon intensity throughout the day.

Time shifting only can lead to carbon emissions reduction in some cases. However, this

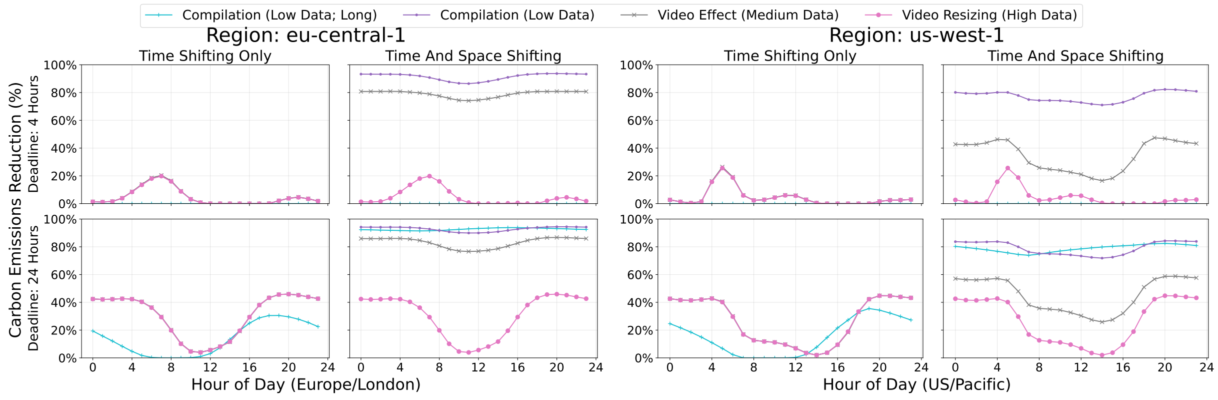


Figure 4.5. Comparison of time shifting vs. space shifting for two different datacenters. Time shifting is most effective for shorter jobs with long deadlines. Space and time shifting can match or beat time shifting in every scenario, with up to 90% reduction in some cases.

only holds for jobs that are launched outside of the low-carbon period during the middle of the day. Workloads launched at noon are already running at the optimal time and therefore do not benefit from time shifting. If we limit the deadline to 4 hours (all workloads except "Compilation (Long)" are 2 hours), then we can only delay up to 2 hours from the request time. This reduces the effectiveness of time shifting, the only viable time to time shift is early in the morning, right before the sun comes out. Around $5-6$ AM in both time zones we see the largest reduction. If we increase the deadline to 24 hours after the job is launched then time shifting is more effective, but the benefit is limited to the daily variation in carbon intensity. The longer the workload is, the less effective time shifting is.

Allowing for time and space shifting leads to much better results. In the 4 hour deadline case, space shifting allows us to pick the best datacenter to run in within the deadline. The only workload that it does not help is the high data workload, and this is because this workload is not suitable for migration. Workloads that have transfer costs that eliminate the benefit of migration can still fall back on time shifting, so the performance of time and space shifting is equal to prior work in this case. Workloads that can be migrated however see up to 90% reduced carbon emissions when compared to time shifting. In the 24 hour deadline case, even long running workloads can see reduced carbon emissions, where time shifting is not as effective.

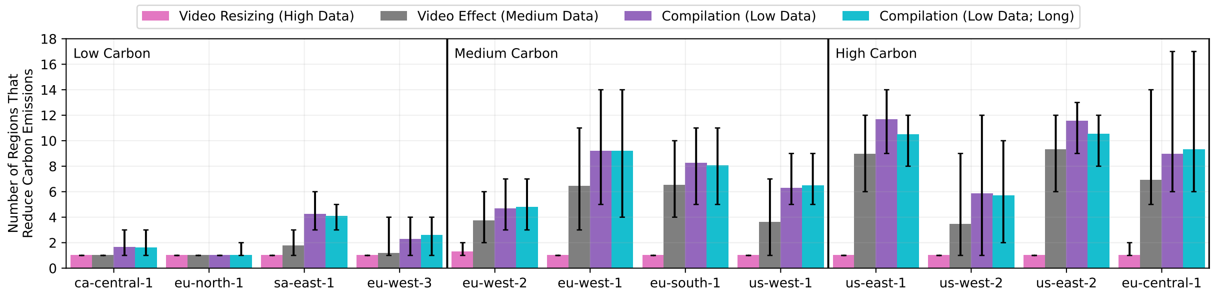


Figure 4.6. Analysis of AWS regions that can benefit from space shifting. For each origin region, the min, average, and max number of regions that lead to an emissions reduction are shown. Regions with high average carbon intensity benefit the most from migration.

This result shows that our method of time and space shifting can significantly reduce carbon emissions further than the state of the art time shifting only technique, especially for low data usage jobs.

4.6.3 Region pair analysis

Workload migration depends both on the job-dependent migration overhead, and the time- and location-dependent CIDT and CIE. These values differ from region to region, and we expect that certain regions can benefit from migration more than others. In this section, we evaluate how many, and what types of regions can benefit. For this analysis, we roughly categorize a selection of AWS datacenters into “low”, “medium”, and “high” carbon intensity, based on the average carbon intensity of the energy grid where the datacenter is located. We simulate our four workloads with a maximum deadline of 24 hours. For the request time, we randomly sample from the year of 2023. For our datacenters available to migrate to, we choose all datacenters in AWS. We then count the number of datacenters which reduce or keep same the carbon emissions (carbon savings of $\geq$0%). When migrating, this number is the number of choices we have that will lead to a carbon savings of $\geq$0%.

We show the average number of datacenters which have $\geq$0% carbon emissions reduction for each of the datacenters in this selection in Figure 4.6. The bars represent the minimum and maximum count of $\geq$0% datacenters in the sample.

We can see in Figure 4.6 that our high data workload almost never benefits from migration. Only one datacenter (the datacenter it is launched in) will lead to $\geq$0% carbon emission reduction. Across all workloads, for low-carbon regions, there are fewer choices for migration. Oftentimes the original region is the best region for the workload. For the medium- and high-carbon regions, there are more choices. Some notable "medium-carbon" regions are the `eu-west-1` and `eu-south-1` regions. Because they are surrounded by low-carbon regions, they still have a large number of choices for datacenters, even though they are not high-carbon. For high-carbon regions, `us-west-2` has much lower minimum values than other regions. This region has an extreme yearly carbon intensity variation, with summer months almost completely comprised of hydropower, and other months using mostly gas. So most of the year, jobs will benefit from migration, however some months it will be best to run most workloads in the original region.

This shows that not all regions benefit from workload migration equally – it depends on both the workload and the carbon intensity of the origin datacenter. In general, high-carbon datacenters benefit more from migration.

4.6.4 Load balancing across regions

If we always send workloads to the datacenter with the lowest carbon intensity, we risk overloading that datacenter's capacity. Because of this, we also evaluate some simple load balancing policies. In the previous Section 4.6.3, we show that for each origin region there will be a set of other regions which will give $\geq$0% carbon emissions reduction. In Figure 4.7 we evaluate three simple load balancing policies.

The first strategy: "Best" is when we always choose the datacenter with the lowest carbon emissions prediction. This strategy leads to the greatest overall carbon emission reduction, however it may not be possible in practice. Since datacenters only have a certain amount of headroom, the lowest carbon datacenter may get more migration requests than it can handle. If each origin region spreads out its migration requests, datacenter headroom is less likely to run out.

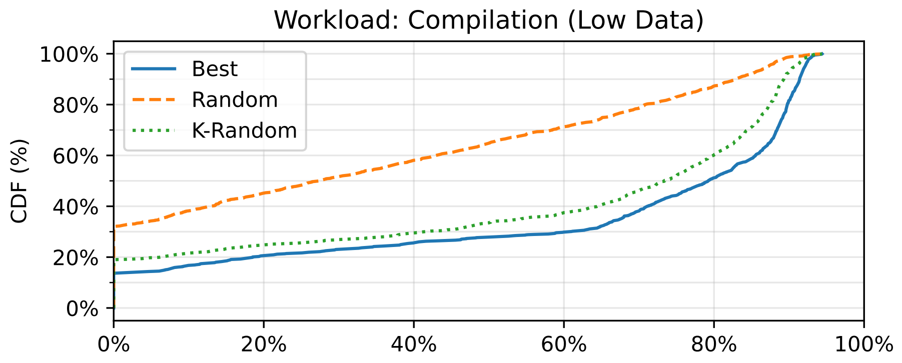


Figure 4.7. Evaluation of some simple load balancing policies. Picking randomly from all eligible regions still leads to some reductions in emissions, however a K-Random ($K = 3$) approach is almost optimal.

The next strategy: “Random” is randomly choosing from the set of regions that have a predicted carbon emission reduction. This spreads the migration requests across a larger pool of candidate regions. Unfortunately, not all regions may have a high carbon emission reduction. In the median case, the carbon emission reduction is only around 35%, compared to the best case median reduction of 80%.

The third strategy: “K-Random” is a compromise by choosing a region from the top N candidate regions. This is shown for $N = 3$ in Figure 4.7. In the median case for this strategy, the emission reduction is not as good as the best case, but only by about 5-10%. Future work can look at other load balancing policies to ensure that datacenters can always maintain a reasonable headroom.

Impact at scale

This result shows that by load balancing across several data centers with low-carbon energy, we can still achieve close to 90% the best case carbon reduction, which is around $70-90\%$ for low-data usage jobs and $30-80\%$ for medium data usage jobs (Section 4.6.2).

If we look at existing capacity study shown in [122], with up to 30% space capacity in data centers, we can achieve up to 25% carbon savings for low-data usage jobs and up to 20% for medium-data usage jobs. Additional data center capacity near renewable sites and/or techniques like turbo-boosting can further increase the carbon saving potentials, although their additional cost (e.g. embodied carbon footprint) may require further study.

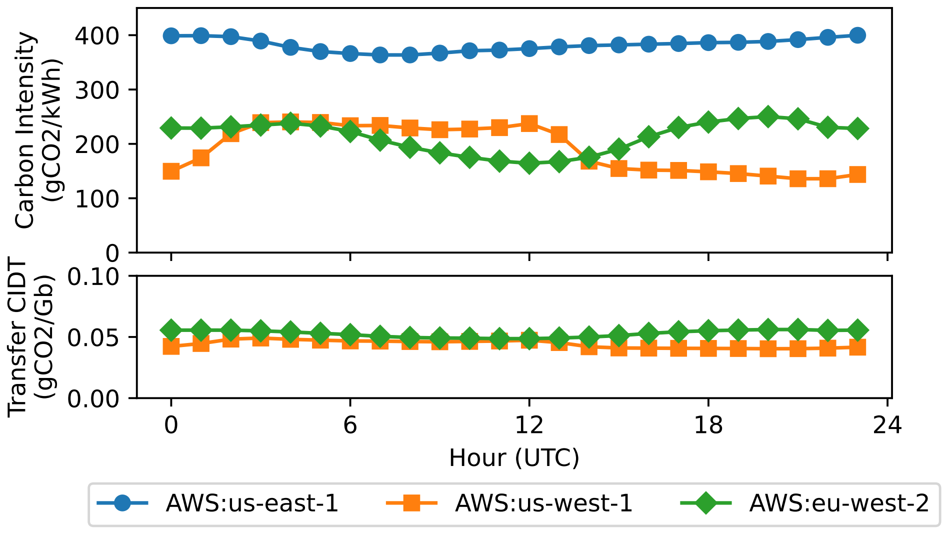
4.6.5 Time analysis

Many datacenters are powered by renewable energy, and many datacenter operators are planning to use more renewable energy to lower overall emissions. However, renewable energy generation is highly time dependent. Solar only works when the sun is shining and wind only works when the wind is blowing. In this section we example what daily and seasonal trends may impact workload migration.

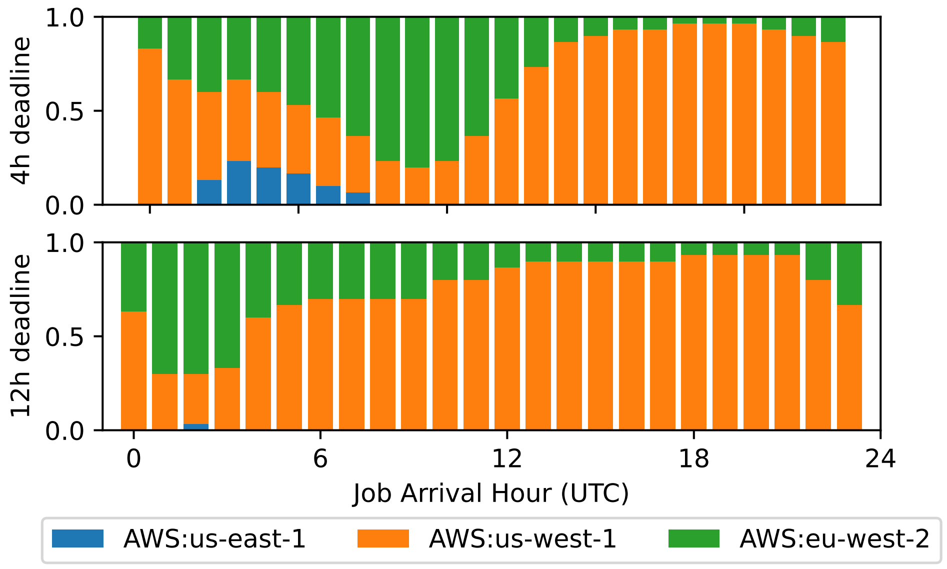
Time of day / Time zone

One downside of solar is that it cannot generate electricity during the night. However, when it's night in California, the sun is shining on the other side of the world. In this section we look at the potential to shift jobs across time zones to take advantage of where the sun is currently shining.

We consider an AWS source location in a high carbon region: `us-east-1` in Virginia, and two solar-heavy regions with daily bathtub curve, AWS `us-west-1` in California, and AWS `eu-west-2` in Great Britain. Both regions experience daily variation due to solar being available only during daytime, but are offsetted in time because of the time zone difference. The carbon intensities of both datacenters are shown at the top of Figure 4.8a. The `eu-west-2` datacenter has the lowest carbon intensity around 12 h UTC (12 h local time), whereas `us-west-1` has the lowest carbon intensity offset by about 10 hours, around 21 h UTC (13 h local time). We also include CIDTs of the paths from the source location to both of these locations at the bottom of Figure 4.8a. The CIDTs of both paths are comparable, since the higher energy cost of submarine cable and the shorter length of the path from `us-east-1` to `eu-west-2` cancel out each other; the curve is mostly flat due to a large number of non-renewable-based regions that the paths traverse through.



(a) Carbon intensity and CIDT time series



(b) Optimal region distribution of medium data job (video effect)

Figure 4.8. The optimal region distribution between datacenters in different time zones. Migrating across time zones allows the utilization of solar energy almost 24 hours a day.

We simulate a medium data-usage job launched in the `us-east-1` datacenter. The job request time is swept across June 2023 in 1 hour increments. We also set the maximum deadline to 4 hours and 12 hours. For each hour of day, we aggregate the optimal datacenter region over the month and show the distribution in Figure 4.8. As we can see, the optimal datacenter region changes based on the hour of the day. When carbon intensity is lowest in `us-west-1`, that datacenter is favored, however when it is night in California, `eu-west-2` is favored. Consequently, given a deadline of h hours, jobs that arrive around $(h-2)$ hours before 12 h UTC favor `eu-west-2`, whereas jobs that arrive $(h-2)$ hours before 21 h UTC favor `us-west-1`.

This shows that we can take advantage of this time zone difference to reduce carbon emissions acroos almost all 24 hours of a day, which is not always possible with time shifting only solutions. The geographically distributed nature of datacenters allows us to place workloads according to where renewable energy is found at any given time.

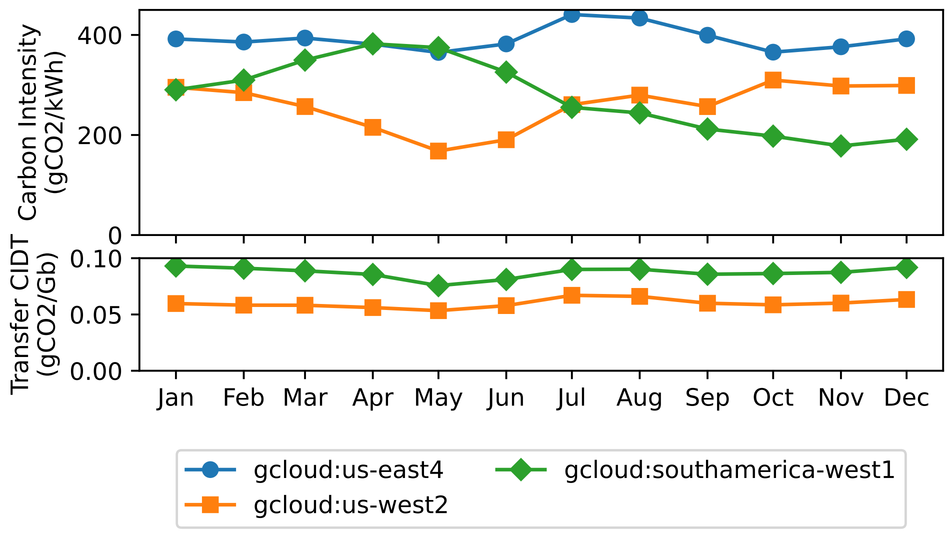
Seasonal

Another important time-varying characteristic in renewable energy is the season. We know from [111] that there are seasonal variations in the carbon intensity of solar-heavy regions. In this section we re-examine how these seasonal changes impact space shifting.

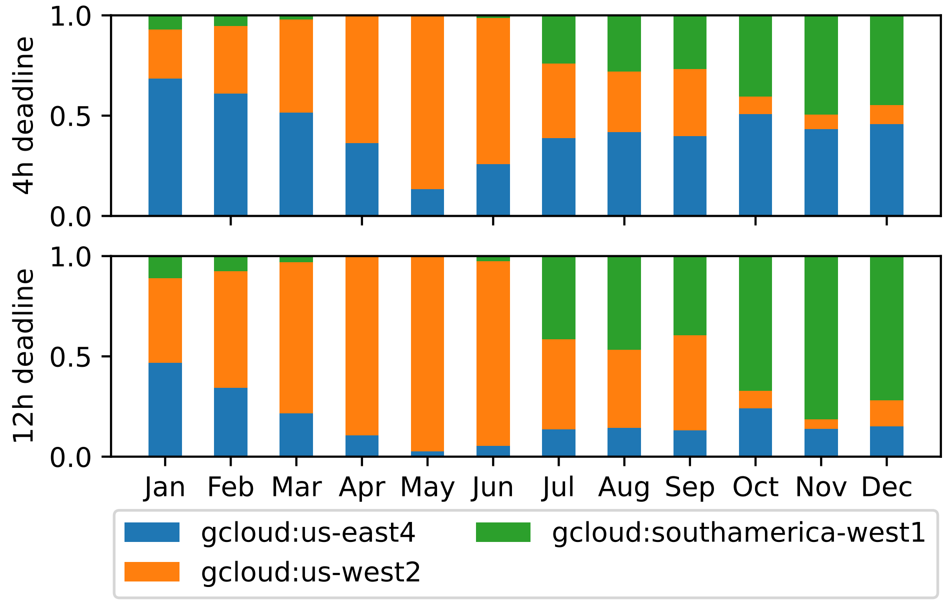
Here, we consider a Google Cloud source location in a high carbon region, `us-east4` in Virginia, and consider two alternative regions with different seasonal patterns: `us-west2` in California, and `southamerica-west1` in Chile. We sampled the job request time at every hour across all days in 2023, and considered both a short deadline of 4 hours and a long deadline of 12 hours, for our medium data workload.

We plotted in Figure 4.9a the monthly average carbon intensity in all three regions, and the monthly average CIDT from source to both alternative regions. In this case, California has lowest average carbon intensity around May, and Chile has the lowest average carbon intensity around November. The CIDT of Virginia to Chile is generally higher than that of Virginia to California, due to the physical longer distance and use of submarine cables, but again the variations are small due to the large number of non-renewable-based regions that the paths traverse through.

In the bottom plot Figure 4.9b, we plotted the distribution of the optimal region across all



(a) Carbon intensity and CIDT time series



(b) Optimal region distribution of medium data job (video effect)

Figure 4.9. The optimal region distribution across seasonal variations. In the southern hemisphere, solar energy is more productive in the winter months, the opposite of the northern hemisphere. The optimal region changes based on this effect.

months, for our medium data usage job (video effect). We can see that out of the two alternative regions, generally the optimal location aligns with the low carbon intensity months. Although in many cases, due to the non-trivial amount of data that needs to be moved, this job might not benefit from migration, and thus stays in Virginia. Comparing the two different deadlines, we observe that workload migration can benefit from a longer deadline, since both regions are mainly powered by solar energy and thus exhibits a daily shift in their carbon intensity. Thus, a longer deadline means that the job is more likely to run at the low carbon period of the day in these regions.

4.7 Limitations and future directions

Having outlined our overall approach, we now discuss its limitations and potential future directions.

4.7.1 Limitations

Carbon intensity granularity:

Due to limitations in the transmission capabilities within an ISO, energy cannot necessarily freely flow throughout the ISO region, and there can be localized "congestion" based on transmission line limitations. In this work, we ignore these localized regions of stranded energy and instead rely on the reported carbon intensity at the ISO level. In general, better energy accounting within the datacenter would further help to select datacenter locations with greater potential for carbon savings.

Capacity constraints:

In our work, we did not consider the effect of network bandwidth limitations for wide-area migrations, nor did we take into account estimates of spare capacity within datacenters to ensure that workloads can "fit" into target datacenter locations.

4.7.2 Future directions

Our study shows that there is a significant potential for carbon savings through shifting workloads geographically. However, there are several challenges and open problems that remain to be addressed to better adopt this approach in practice.

We can envision an approach where application states are geo-replicated to ensure that the needed data is available in regions with complimentary low-carbon energy availability. As an example, replicas placed sufficiently far apart that solar power is available at at least one replica location. Further, we can imagine using our dataset to better inform hardware placement by putting GPUs, TPUs, etc in regions which are likely to have jobs migrated to them. Finally, we see an opportunity to further develop carbon-aware load balancing approaches that select datacenters with sufficient resources to accept migrations from high-carbon regions.

4.8 Summary

In this chapter, we show the benefits of geographical space shifting to reduce the overall carbon impact of workloads with more relaxed completion time requirements. Similar to previous studies, we have shown that the carbon cost of WAN transfers is significant, and needs to be taken into account when choosing locations for geographic load shifting. We outlined an approach and methodology for increasing the fidelity of WAN path carbon footprint estimation, and have shown that with such an estimate it is possible to reduce the carbon footprint of many workloads significantly.

This chapter, together with Chapter 3, shows that carbon-aware workload migration or space-shifting can indeed be a powerful tool to reduce the carbon footprint of data centers. By utilizing relaxed timelines of computing workloads and catering to the time- and space-varying characteristics of renewable energy, we can significantly reduce the carbon footprint of data centers across a wide range of applications, helping to achieve the urgent goal of a carbon-neutral cloud.

4.9 Acknowledgements

Chapter 4 in part is currently being prepared for submission for publication of the material. Yibo Guo, Amanda Tomlinson, Runlong Su, and George Porter, 2024. The dissertation author

was the primary investigator and author of this paper.

Chapter 5

Conclusion

Data centers are a crucial infrastructure powering the modern digital economy, and are likely continuing to grow in the years and decades to come. As the scale and global footprint of data centers keep increasing, the energy consumption and carbon emissions of data centers are becoming important concerns. To ensure the sustainability of data centers, we need to explore all possible ways to improve the energy efficiency and reduce the carbon emissions of data centers.

In this dissertation, I looked at solutions from the networking perspective, both within a data center and across globally distributed data centers.

Within a datacenter, I focus on the energy efficiency of intra-data center networks, or DCNs. DCNs are becoming a crucial infrastructure in modern data centers and its energy footprint has grown beyond single-digit percentage. As bandwidth demand continues to grow, the cost of relying on incremental improvement in the underlying switch chips no longer suffice. In this dissertation, I explored how to improve the efficiency and scalability of DCNs by better engineering traditional packet switched networks.

In the second part of this dissertation, I focus on the environmental impact of data centers in operational carbon emissions, which has become a more pressing concern due to their large energy footprint and urgent climate goals. I explore in depth the approach of carbon-aware space-shifting of workloads across globally distributed data centers, as an alternative way to adopt low-carbon renewable energy in data center operations. This study shows that although the carbon cost of workload migration over the wide area is much higher than existing study shows, by selecting the right set of jobs and optimizing for the combined carbon cost of both computation and network, we can still achieve significant carbon reduction over the alternative time-shifting approach.

By addressing the energy efficiency and carbon emissions of data centers from the networking perspective, this dissertation provides a dual-pronged approach to improve the sustainability of data centers.

Future data centers will continue to grow in scale, demanding continuous improvement in both energy efficiency and carbon emission reduction. Driving energy efficiency has been a long-standing goal since it is directly tied to the operational cost of data centers. However, to meet the urgent climate goals, we need to re-examine how we evaluate the "cost" of computing and optimize for the carbon cost of computing, not just the dollar cost. Furthermore, careful accounting of the full lifecycle carbon cost of computing equipment is needed to balance between operational and embodied carbon emissions.

Bibliography

[1] *2020 – The Year that Cloud Service Revenues Finally Dwarfed Enterprise Spending on Data Centers*. Mar. 2021. URL: https://www.srgresearch.com/articles/2020-the-year-that-cloud-service-revenues-finally-dwarfed-enterprise-spending-on-data-centers.

[2] *400G ZR/ZR+ QSFP-DD-DCO — Lumentum Operations LLC*. 2024. URL: https://www.lumentum.com/en/products/400g-zrzr-qsfp-dd-dco.

[3] *802.1QBP - equal cost multiple paths*. Mar. 2014. URL: https://www.ieee802.org/1/pages/802.1bp.html.

[4] Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Kiwan Maeng, Udit Gupta, Manoj Chakkaravarthy, David Brooks, and Carole-Jean Wu. "Carbon Explorer: A Holistic Framework for Designing Carbon Aware Datacenters". In: *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ASPLOS 2023. Vancouver, BC, Canada: Association for Computing Machinery, 2023, pp. 118–132. ISBN: 9781450399166. DOI: 10.1145/3575693.3575754. URL: https://doi.org/10.1145/3575693.3575754.

[5] Alexandru Agache, Razvan Deaconescu, and Costin Raiciu. "Increasing Datacenter Network Utilisation with GRIN". In: *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*. NSDI'15. Oakland, CA: USENIX Association, 2015, pp. 29–42. ISBN: 9781931971218.

[6] Anup Agarwal, Jinghan Sun, Shadi Noghabi, Srinivasan Iyengar, Anirudh Badam, Ranveer Chandra, Srinivasan Seshan, and Shivkumar Kalyanaraman. "Redesigning Data

Centers for Renewable Energy”. In: *HotNets*. 2021. URL: https://www.microsoft.com/en-us/research/publication/redesigning-data-centers-for-renewable-energy/.

[7] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. “Data Center TCP (DCTCP)”. In: *Proceedings of the ACM SIGCOMM 2010 Conference*. SIGCOMM ’10. New Delhi, India: Association for Computing Machinery, 2010, pp. 63–74. ISBN: 9781450302012. DOI: 10.1145/1851182.1851192. URL: https://doi.org/10.1145/1851182.1851192.

[8] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. “PFabric: Minimal near-Optimal Datacenter Transport”. In: *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*. SIGCOMM ’13. Hong Kong, China: Association for Computing Machinery, 2013, pp. 435–446. ISBN: 9781450320566. DOI: 10.1145/2486001.2486031. URL: https://doi.org/10.1145/2486001.2486031.

[9] Ethernet Alliance. *The 2020 Ethernet Roadmap*. Feb. 2020. URL: https://ethernetalliance.org/technology/2020-roadmap/ (visited on 01/24/2021).

[10] Ethernet Alliance. *The 2024 Ethernet Roadmap*. Mar. 2024. URL: https://ethernetalliance.org/technology/ethernet-roadmap/ (visited on 03/01/2024).

[11] Scott Anderson, Loqman Salamatian, Zachary S Bischof, Alberto Dainotti, and Paul Barford. “iGDB: connecting the physical and logical layers of the internet”. In: *Proceedings of the 22nd ACM Internet Measurement Conference*. 2022, pp. 433–448.

[12] Alexey Andreyev, Xu Wang, and Alex Eckert. *Reinventing our data center network with F16, Minipack*. Mar. 2019. URL: https://engineering.fb.com/data-center-engineering/f16-minipack/.

[13] Lori Aniti. *California's curtailments of solar electricity generation continue to increase - U.S. Energy Information Administration (EIA)*. Aug. 2021. URL: https://www.eia.gov/todayinenergy/detail.php?id=49276.

[14] Apple Security Engineering and Architecture, User Privacy, Core Operating Systems, Services Engineering, and Machine Learning and AI. *Blog - Private Cloud Compute: A new frontier for AI privacy in the cloud - Apple Security Research*. June 2024. URL: https://security.apple.com/blog/private-cloud-compute/.

[15] Yuval Bachar. *Introducing "6-pack": the first open hardware modular switch*. June 2018. URL: https://engineering.fb.com/2015/02/11/production-engineering/introducing-6-pack-the-first-open-hardware-modular-switch/.

[16] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, and Hugh Williams. "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*. SIGCOMM '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 782–797. ISBN: 9781450379557. DOI: 10.1145/3387514.3406221. URL: https://doi.org/10.1145/3387514.3406221.

[17] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. *The datacenter as a computer: Designing warehouse-scale machines*. Springer Nature, 2019.

[18] Aaron Bloom, Josh Novacheck, Greg Brinkman, James McCalley, Armando Figueroa-Acevedo, Ali Jahanbani-Ardakani, Hussam Nosair, Abhinav Venkatraman, Jay Caspary, Dale Osborn, and Jessica Lau. "The Value of Increased HVDC Capacity Between Eastern and Western U.S. Grids: The Interconnections Seam Study". In: *IEEE Transactions on Power Systems* 37.3 (2022), pp. 1760–1769. DOI: 10.1109/TPWRS.2021.3115092.

[19] Brad Smith. *Microsoft will be carbon negative by 2030 - The Official Microsoft Blog*. Jan. 2020. URL: https://blogs.microsoft.com/blog/2020/01/16/microsoft-will-be-carbon-negative-by-2030/ (visited on 07/01/2024).

[20] Qizhe Cai, Shubham Chaudhary, Midhul Vuppalapati, Jaehyun Hwang, and Rachit Agarwal. "Understanding Host Network Stack Overheads". In: *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. SIGCOMM '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 65–77. ISBN: 9781450383837. DOI: 10.1145/3452296.3472888. URL: https://doi.org/10.1145/3452296.3472888.

[21] Charles Clos. "A study of non-blocking switching networks". In: *The Bell System Technical Journal* 32.2 (1953), pp. 406–424. DOI: 10.1002/j.1538-7305.1953.tb01433.x.

[22] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. "OSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility". In: *IEEE/ACM Transactions on Networking* 22.2 (2014), pp. 498–511. DOI: 10.1109/TNET.2013.2253120.

[23] Li Chen, Kai Chen, Zhonghua Zhu, Minlan Yu, George Porter, Chunming Qiao, and Shan Zhong. "Enabling Wide-Spread Communications on Optical Fabric with MegaSwitch". In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA: USENIX Association, Mar. 2017, pp. 577–593. ISBN: 978-1-931971-37-9. URL: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/chen.

[24] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. "One Trillion Edges: Graph Processing at Facebook-Scale". In: *Proc. VLDB Endow.* 8.12 (Aug. 2015), pp. 1804–1815. ISSN: 2150-8097. DOI: 10.14778/2824032.2824077. URL: https://doi.org/10.14778/2824032.2824077.

[25] *Data Centers and Servers — Department of Energy*. 2024. URL: https://www.energy.gov/eere/buildings/data-centers-and-servers.

[26] Alex de Vries. “The growing energy footprint of artificial intelligence”. In: *Joule* 7.10 (2023), pp. 2191–2194. ISSN: 2542-4351. DOI: https://doi.org/10.1016/j.joule.2023.09.004. URL: https://www.sciencedirect.com/science/article/pii/S2542435123003653.

[27] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. In: *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*. OSDI’04. San Francisco, CA: USENIX Association, 2004, pp. 137–149. URL: http://static.usenix.org/event/osdi04/tech/full_papers/dean/dean.pdf.

[28] P. Dong, W. Yang, K. Xue, W. Tang, K. Gao, and J. Huang. “Tuning the Aggressive Slow-Start Behavior of MPTCP for Short Flows”. In: *IEEE Access* 7 (2019), pp. 6010–6024. DOI: 10.1109/ACCESS.2018.2889339.

[29] Pingping Dong, Wenjun Yang, Wensheng Tang, Jiawei Huang, Haodong Wang, Yi Pan, and Jianxin Wang. “Reducing transport latency for short flows with multipath TCP”. In: *Journal of Network and Computer Applications* 108 (2018), pp. 20–36. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2018.02.005. URL: http://www.sciencedirect.com/science/article/pii/S1084804518300420.

[30] Hewlett Packard Enterprise. *HPE Ethernet 4x25Gb 1-port 620QSFP28 Adapter - Overview*. 2021. URL: https://support.hpe.com/hpesc/public/docDisplay?docId=emr_na-c05220334.

[31] Y. Shaya Fainman, Joseph Ford, William M. Mellette, Shayan Mookherjea George Porter, Alex C. Snoeren, George Papen, Saman Saeedi, John Cunningham, Ashok Krishnamoorthy, Michael Gehl, Christopher T. DeRose, Paul S. Davids, Douglas C. Trotter, Andrew L. Starbuck, Christina M. Dallo, Dana Hood, Andrew Pomerene, and Anthony Lentine. “LEED: A Lightwave Energy-Efficient Datacenter”. In: *2019 Optical Fiber Communications Conference and Exhibition (OFC)*. 2019, pp. 1–3.

[32] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. “A Scalable, Commodity Data Center Network Architecture”. In: *Proceedings of the ACM SIGCOMM 2008*

Conference on Data Communication. SIGCOMM '08. Seattle, WA, USA: Association for Computing Machinery, 2008, pp. 63–74. ISBN: 9781605581750. DOI: 10.1145/1402958.1402967. URL: https://doi.org/10.1145/1402958.1402967.

[33] Nathan Farrington, Alex Forencich, George Porter, P.-C. Sun, Joseph E. Ford, Yeshaiahu Fainman, George C. Papen, and Amin Vahdat. "A Multiport Microsecond Optical Circuit Switch for Data Center Networking". In: *IEEE Photonics Technology Letters* 25.16 (2013), pp. 1589–1592. DOI: 10.1109/LPT.2013.2270462.

[34] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. "Helios: a hybrid electrical/optical switch architecture for modular data centers". In: *SIGCOMM Comput. Commun. Rev.* 40.4 (Aug. 2010), pp. 339–350. ISSN: 0146-4833. DOI: 10.1145/1851275.1851223. URL: https://doi.org/10.1145/1851275.1851223.

[35] Alex Forencich, Alex C. Snoeren, George Porter, and George Papen. "Corundum: An Open-Source 100-Gbps Nic". In: *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 2020, pp. 38–46.

[36] Peter X. Gao, Akshay Narayan, Gautam Kumar, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. "PHost: Distributed near-Optimal Datacenter Transport over Commodity Network Fabric". In: *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. CoNEXT '15. Heidelberg, Germany: Association for Computing Machinery, 2015. ISBN: 9781450334129. DOI: 10.1145/2716281.2836086. URL: https://doi.org/10.1145/2716281.2836086.

[37] Peter Xiang Gao, Andrew R. Curtis, Bernard Wong, and Srinivasan Keshav. "It's not easy being green". In: *SIGCOMM Comput. Commun. Rev.* 42.4 (Aug. 2012), pp. 211–222. ISSN: 0146-4833. DOI: 10.1145/2377677.2377719. URL: https://doi.org/10.1145/2377677.2377719.

[38] Clarissa Garcia. *Data Center Energy Use - AKCP monitoring*. Sept. 2023. URL: https://www.akcp.com/blog/the-real-amount-of-energy-a-data-center-use/.

[39] Manya Ghobadi, Ashkan Seyedi, Chongjin Xie, Hong Liu, and Rob Stone. "ACM SIGCOMM 2021 Workshop on Optical Systems: Industrial Panel". Aug. 2021. URL: https://conferences.sigcomm.org/sigcomm/2021/workshop-optsys.html.

[40] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel Kilper. "ProjecToR: Agile Reconfigurable Data Center Interconnect". In: *Proceedings of the 2016 ACM SIGCOMM Conference*. SIGCOMM '16. Florianopolis, Brazil: Association for Computing Machinery, 2016, pp. 216–229. ISBN: 9781450341936. DOI: 10.1145/2934872.2934911. URL: https://doi.org/10.1145/2934872.2934911.

[41] Íñigo Goiri, Kien Le, Thu D. Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. "GreenHadoop: Leveraging Green Energy in Data-Processing Frameworks". In: *Proceedings of the 7th ACM European Conference on Computer Systems*. EuroSys '12. Bern, Switzerland: Association for Computing Machinery, 2012, pp. 57–70. ISBN: 9781450312233. DOI: 10.1145/2168836.2168843. URL: https://doi.org/10.1145/2168836.2168843.

[42] Google. *Aiming to Achieve Net-Zero Emissions - Google Sustainability*. 2024. URL: https://sustainability.google/operating-sustainably/net-zero-carbon/ (visited on 07/01/2024).

[43] *Google code archive - long-term storage for Google code project hosting*. 2021. URL: https://code.google.com/archive/p/k-shortest-paths/.

[44] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. "VL2: A Scalable and Flexible Data Center Network". In: *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*. SIGCOMM '09. Barcelona, Spain: Association for

Computing Machinery, 2009, pp. 51–62. ISBN: 9781605585949. DOI: 10.1145/1592568.1592576. URL: https://doi.org/10.1145/1592568.1592576.

[45] *Greenhouse Gases Equivalencies Calculator - Calculations and References*. Feb. 2024. URL: https://www.epa.gov/energy/greenhouse-gases-equivalencies-calculator-calculations-and-references.

[46] Yibo Guo, William M. Mellette, Alex C. Snoeren, and George Porter. “Scaling beyond packet switch limits with multiple dataplanes”. In: *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies*. CoNEXT ’22. Roma, Italy: Association for Computing Machinery, 2022, pp. 214–231. ISBN: 9781450395083. DOI: 10.1145/3555050.3569141. URL: https://doi.org/10.1145/3555050.3569141.

[47] Yibo Guo and George Porter. “A metric for factoring data movement into chasing the sun”. 1st Workshop on NetZero Carbon Computing. Montreal, Canada, Feb. 2023. URL: https://netzero.sysnet.ucsd.edu/program/.

[48] Yibo Guo and George Porter. “Carbon-aware inter-datacenter workload scheduling and placement”. Poster session of 20th USENIX Symposium on Networked Systems Design and Implementation. Boston, MA, Apr. 2023. URL: https://www.usenix.org/conference/nsdi23/poster-session.

[49] Yibo Guo, Amanda Tomlinson, Runlong Su, and George Porter. “The effect of the network in cutting carbon for geo-shifted workloads”. In submission. 2024.

[50] Walid A. Hanafy, Qianlin Liang, Noman Bashir, David Irwin, and Prashant Shenoy. “CarbonScaler: Leveraging Cloud Workload Elasticity for Optimizing Carbon-Efficiency”. In: *Proc. ACM Meas. Anal. Comput. Syst.* 7.3 (Dec. 2023). DOI: 10.1145/3626788. URL: https://doi.org/10.1145/3626788.

[51] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W. Moore, Gianni Antichi, and Marcin Wójcik. “Re-Architecting Datacenter Networks and Stacks for Low Latency and High Performance”. In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. SIGCOMM ’17. Los Angeles, CA, USA: Association for Computing Machinery, 2017, pp. 29–42. ISBN: 9781450346535. DOI: 10.1145/3098822.3098825. URL: https://doi.org/10.1145/3098822.3098825.

[52] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, James Law, Kevin Lee, Jason Lu, Pieter Noordhuis, Misha Smelyanskiy, Liang Xiong, and Xiaodong Wang. “Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective”. In: *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2018, pp. 620–629. DOI: 10.1109/HPCA.2018.00059.

[53] IEA. *Electricity 2024 – analysis*. Jan. 2024. URL: https://www.iea.org/reports/electricity-2024.

[54] IEA. *Emissions – global energy & CO2 status report 2019 – analysis*. Mar. 2019. URL: https://www.iea.org/reports/global-energy-co2-status-report-2019/emissions.

[55] *Intel® Xeon® Processor E5-2640 v4*. Feb. 2024. URL: https://ark.intel.com/content/www/us/en/ark/products/92984/intel-xeon-processor-e5-2640-v4-25m-cache-2-40-ghz.html.

[56] Paras Jain, Sam Kumar, Sarah Wooders, Shishir G. Patil, Joseph E. Gonzalez, and Ion Stoica. “Skyplane: Optimizing Transfer Cost and Throughput Using Cloud-Aware Overlays”. In: *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 1375–1389. ISBN: 978-1-939133-33-5. URL: https://www.usenix.org/conference/nsdi23/presentation/jain.

[57] Aled James and Daniel Schien. “A Low Carbon Kubernetes Scheduler”. In: *ICT4S*. 2019.

[58] Vimalkumar Jeyakumar, Mohammad Alizadeh, David Mazières, Balaji Prabhakar, Albert Greenberg, and Changhoon Kim. "EyeQ: Practical Network Performance Isolation at the Edge". In: *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX Association, Apr. 2013, pp. 297–311. ISBN: 978-1-931971-00-3. URL: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/jeyakumar.

[59] Norman P. Jouppi, Doe Hyun Yoon, George Kurian, Sheng Li, Nishant Patil, James Laudon, Cliff Young, and David Patterson. "A Domain-Specific Supercomputer for Training Deep Neural Networks". In: *Commun. ACM* 63.7 (June 2020), pp. 67–78. ISSN: 0001-0782. DOI: 10.1145/3360307. URL: https://doi.org/10.1145/3360307.

[60] Bran Knowles. "ACM TechBrief: Computing and Climate Change". In: (2021).

[61] *Lancium's energy technology and infrastructure solutions deliver the lowest cost, green energy at scale.* Apr. 2024. URL: https://lancium.com/solutions/.

[62] D. Li and J. Wu. "On the design and analysis of Data Center Network architectures for interconnecting dual-port servers". In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 2014, pp. 1851–1859.

[63] Liuzixuan Lin and Andrew A Chien. "Adapting Datacenter Capacity for Greener Datacenters and Grid". In: *Proceedings of the 14th ACM International Conference on Future Energy Systems*. e-Energy '23. Orlando, FL, USA: Association for Computing Machinery, 2023, pp. 200–213. ISBN: 9798400700323. DOI: 10.1145/3575813.3595197. URL: https://doi.org/10.1145/3575813.3595197.

[64] He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter. "Circuit Switching Under the Radar with REACToR". In: *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. Seattle, WA: USENIX Association, Apr. 2014, pp. 1–15.

ISBN: 978-1-931971-09-6. URL: https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/liu_he.

[65] Hong Liu. *ACM SIGCOMM 2021 Workshop on Optical Systems invited talk: The Evolving Role of Optics for Datacenter Network and Machine Learning*. Aug. 2021. URL: https://conferences.sigcomm.org/sigcomm/2021/workshop-optsys.html.

[66] Vincent Liu, Danyang Zhuo, Simon Peter, Arvind Krishnamurthy, and Thomas Anderson. "Subways: A Case for Redundant, Inexpensive Data Center Edge Links". In: *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. CoNEXT '15. Heidelberg, Germany: Association for Computing Machinery, 2015. ISBN: 9781450334129. DOI: 10.1145/2716281.2836112. URL: https://doi.org/10.1145/2716281.2836112.

[67] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H. Low, and Lachlan L.H. Andrew. "Greening geographical load balancing". In: *SIGMETRICS Perform. Eval. Rev.* 39.1 (June 2011), pp. 193–204. ISSN: 0163-5999. DOI: 10.1145/2007116.2007139. URL: https://doi.org/10.1145/2007116.2007139.

[68] Gurobi Optimization LLC. *Gurobi Optimizer*. 2021. URL: https://www.gurobi.com/products/gurobi-optimizer/ (visited on 01/24/2021).

[69] Diptyaroop Maji, Prashant Shenoy, and Ramesh K. Sitaraman. "CarbonCast: multi-day forecasting of grid carbon intensity". In: *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. BuildSys '22. Boston, Massachusetts: Association for Computing Machinery, 2022, pp. 198–207. ISBN: 9781450398909. DOI: 10.1145/3563357.3564079. URL: https://doi.org/10.1145/3563357.3564079.

[70] Electricity Maps. *2023 Hourly Carbon Intensity Data. Electricity Maps Data Portal, ver. January 17, 2024*. 2024.

[71] *MaxMind GeoLite2 Free Geolocation Data*. Jan. 2024. URL: https://dev.maxmind.com/geoip/geolite2-free-geolocation-data.

[72] William M. Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C. Snoeren, and George Porter. "Expanding across time to deliver bandwidth efficiency and low latency". In: *17th USENIX Symposium on Networked Systems Design and Implementation*. NSDI '20. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 1–18. ISBN: 978-1-939133-13-7. URL: https://www.usenix.org/conference/nsdi20/presentation/mellette.

[73] William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. "RotorNet: A Scalable, Low-complexity, Optical Datacenter Network". In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. SIGCOMM '17. Los Angeles, CA, USA: Association for Computing Machinery, 2017, pp. 267–280. ISBN: 9781450346535. DOI: 10.1145/3098822.3098838. URL: https://doi.org/10.1145/3098822.3098838.

[74] William M. Mellette, Alex C. Snoeren, and George Porter. "P-FatTree: A multi-channel datacenter network topology". In: *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. HotNets '16. Atlanta, GA, USA: Association for Computing Machinery, 2016, pp. 78–84. ISBN: 9781450346610. DOI: 10.1145/3005745.3005746. URL: https://doi.org/10.1145/3005745.3005746.

[75] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. "PipeDream: Generalized Pipeline Parallelism for DNN Training". In: *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. SOSP '19. Huntsville, Ontario, Canada: Association for Computing Machinery, 2019, pp. 1–15. ISBN: 9781450368735. DOI: 10.1145/3341301.3359646. URL: https://doi.org/10.1145/3341301.3359646.

[76] Arista Networks. *Arista 7368X4 Series*. Apr. 2019. URL: https://www.arista.com/en/products/7368x4-series.

[77] Tess Ferrandez Norlander and Anders Lybecker. *Saving CO2 using location and time shifting in Azure - ISE Developer Blog*. Nov. 2023. URL: https://devblogs.microsoft.com/ise/saving-co2-using-location-and-time-shifting-in-azure/.

[78] *PeeringDB*. 2024. URL: https://www.peeringdb.com/.

[79] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. "Fastpass: A Centralized "Zero-Queue" Datacenter Network". In: *Proceedings of the 2014 ACM Conference on SIGCOMM*. SIGCOMM '14. Chicago, Illinois, USA: Association for Computing Machinery, 2014, pp. 307–318. ISBN: 9781450328364. DOI: 10.1145/2619239.2626309. URL: https://doi.org/10.1145/2619239.2626309.

[80] Nadja Popovich and Brad Plumer. June 2023. URL: https://www.nytimes.com/interactive/2023/06/12/climate/us-electric-grid-energy-transition.html.

[81] Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, and Amin Vahdat. "Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Networking". In: *Proceedings of the ACM SIGCOMM 2022 Conference*. SIGCOMM '22. Amsterdam, Netherlands: Association for Computing Machinery, 2022, pp. 66–85. ISBN: 9781450394208. DOI: 10.1145/3544216.3544265. URL: https://doi.org/10.1145/3544216.3544265.

[82] Ana Radovanovic. *Our data centers now work harder when the sun shines and Wind Blows*. Apr. 2020. URL: https://blog.google/inside-google/infrastructure/data-centers-work-harder-sun-shines-wind-blows/.

[83] Ana Radovanovic, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. "Carbon-aware computing for datacenters". In: *arXiv preprint arXiv:2106.11750* (2021).

[84] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, Saurav Talukdar, Eric Mullen, Kendal Smith, MariEllen Cottman, and Walfredo Cirne. "Carbon-Aware Computing for Datacenters". In: *IEEE Transactions on Power Systems* 38.2 (2023), pp. 1270–1280. DOI: 10.1109/TPWRS.2022.3173250.

[85] *Regions and zones — Compute Engine Documentation — Google Cloud*. Nov. 2023. URL: https://cloud.google.com/compute/docs/regions-zones.

[86] *Regions, Availability Zones, and Local Zones - Amazon Relational Database Service*. Nov. 2023. URL: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html.

[87] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. "Inside the Social Network's (Datacenter) Network". In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. SIGCOMM '15. London, United Kingdom: Association for Computing Machinery, 2015, pp. 123–137. ISBN: 9781450335423. DOI: 10.1145/2785956.2787472. URL: https://doi.org/10.1145/2785956.2787472.

[88] Jennifer Runyon. Apr. 2021. URL: https://www.power-grid.com/td/transmission/the-us-needs-a-macrogrid-to-move-electricity-from-areas-that-make-it-to-areas-that-need-it/#gref.

[89] *S Series Optical Circuit Switch — CALIENT Technologies*. 2021. URL: https://www.calient.net/products/s-series-photonic-switch/.

[90] Zhiming Shen, Qin Jia, Gur-Eyal Sela, Ben Rainero, Weijia Song, Robbert van Renesse, and Hakim Weatherspoon. "Follow the Sun through the Clouds: Application Migration for Geographically Shifting Workloads". In: *Proceedings of the Seventh ACM Symposium on Cloud Computing*. SoCC '16. Santa Clara, CA, USA: Association for Computing

Machinery, 2016, pp. 141–154. ISBN: 9781450345255. DOI: 10.1145/2987550.2987561. URL: https://doi.org/10.1145/2987550.2987561.

[91] Zhiming Shen, Qin Jia, Gur-Eyal Sela, Ben Rainero, Weijia Song, Robbert van Renesse, and Hakim Weatherspoon. "Follow the Sun through the Clouds: Application Migration for Geographically Shifting Workloads". In: *Proceedings of the Seventh ACM Symposium on Cloud Computing*. SoCC '16. Santa Clara, CA, USA: Association for Computing Machinery, 2016, pp. 141–154. ISBN: 9781450345255. DOI: 10.1145/2987550.2987561. URL: https://doi.org/10.1145/2987550.2987561.

[92] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network". In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. SIGCOMM '15. London, United Kingdom: Association for Computing Machinery, 2015, pp. 183–197. ISBN: 9781450335423. DOI: 10.1145/2785956.2787508. URL: https://doi.org/10.1145/2785956.2787508.

[93] Rachee Singh, Nikolaj Bjorner, Sharon Shoham, Yawei Yin, John Arnold, and Jamie Gaudette. "Cost-Effective Capacity Provisioning in Wide Area Networks with Shoofly". In: *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. SIGCOMM '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 534–546. ISBN: 9781450383837. DOI: 10.1145/3452296.3472895. URL: https://doi.org/10.1145/3452296.3472895.

[94] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. "Jellyfish: Networking Data Centers Randomly". In: *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX,

2012, pp. 225–238. ISBN: 978-931971-92-8. URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/singla.

[95] Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, and Prashant Shenoy. “Ecovisor: A Virtual Energy System for Carbon-Efficient Applications”. In: *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ASPLOS 2023. Vancouver, BC, Canada: Association for Computing Machinery, 2023, pp. 252–265. ISBN: 9781450399166. DOI: 10.1145/3575693.3575709. URL: https://doi.org/10.1145/3575693.3575709.

[96] Rob Stone. *ACM SIGCOMM 2021 Workshop on Optical Systems invited talk: Co-packaged Optics in the Data Center*. Aug. 2021. URL: https://conferences.sigcomm.org/sigcomm/2021/workshop-optsys.html.

[97] Seyedali Tabaeiaghdaei, Simon Scherrer, Jonghoon Kwon, and Adrian Perrig. “Carbon-Aware Global Routing in Path-Aware Networks”. In: *Proceedings of the 14th ACM International Conference on Future Energy Systems*. e-Energy ’23. Orlando, FL, USA: Association for Computing Machinery, 2023, pp. 144–158. ISBN: 9798400700323. DOI: 10.1145/3575813.3595192. URL: https://doi.org/10.1145/3575813.3595192.

[98] Ole Tange. *GNU Parallel 2018*. Ole Tange, Apr. 2018. DOI: 10.5281/zenodo.1146014. URL: https://doi.org/10.5281/zenodo.1146014.

[99] *TechBriefs computing and Climate Change - Acm.org*. Nov. 2021. URL: https://www.acm.org/binaries/content/assets/public-policy/techbriefs/computing-and-climate-change-nov-2021.pdf.

[100] *The amount of data center energy use - AKCP monitoring*. Feb. 2022. URL: https://www.akcp.com/blog/the-real-amount-of-energy-a-data-center-use/.

[101] *The CAIDA Macroscopic Internet Topology Data Kit - 2022-02*. Aug. 2023. URL: https://www.caida.org/catalog/datasets/internet-topology-data-kit.

[102] Eric Urban, Cory Fowler, Ingrid Henkel, and Nitin Mehrotra. *What are Azure AI services? - azure AI services*. Mar. 2024. URL: https://learn.microsoft.com/en-us/azure/ai-services/what-are-ai-services.

[103] Amin Vahdat, Mohammad Al-Fares, Nathan Farrington, Radhika Niranjan Mysore, George Porter, and Sivasankar Radhakrishnan. "Scale-Out Networking in the Data Center". In: *IEEE Micro* 30.4 (2010), pp. 29–41. DOI: 10.1109/MM.2010.72.

[104] Asaf Valadarsky, Gal Shahaf, Michael Dinitz, and Michael Schapira. "Xpander: Towards Optimal-Performance Datacenters". In: *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '16. Irvine, California, USA: Association for Computing Machinery, 2016, pp. 205–219. ISBN: 9781450342926. DOI: 10.1145/2999572.2999580. URL: https://doi.org/10.1145/2999572.2999580.

[105] Valiant Communications Limited. *Valiant: Multiplexers, teleprotection, GPS PRC, TDM over IP*. 2024. URL: https://www.valiantcom.com/routers/5040-mpls-router.html.

[106] Ward Van Heddeghem, Filip Idzikowski, Willem Vereecken, Didier Colle, Mario Pickavet, and Piet Demeester. "Power consumption modeling in optical multilayer networks". In: *Photonic Network Communications* 24 (2012), pp. 86–102.

[107] Yves Vanaubel, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. "MPLS Under the Microscope: Revealing Actual Transit Path Diversity". In: *Proceedings of the 2015 Internet Measurement Conference*. IMC '15. Tokyo, Japan: Association for Computing Machinery, 2015, pp. 49–62. ISBN: 9781450338486. DOI: 10.1145/2815675.2815687. URL: https://doi.org/10.1145/2815675.2815687.

[108] Yves Vanaubel, Pascal Mérindol, Jean-Jacques Pansiot, and Benoit Donnet. "Through the Wormhole: Tracking Invisible MPLS Tunnels". In: *Proceedings of the 2017 Internet Measurement Conference*. IMC '17. London, United Kingdom: Association for Computing Machinery, 2017, pp. 29–42. ISBN: 9781450351188. DOI: 10.1145/3131365.3131378. URL: https://doi.org/10.1145/3131365.3131378.

[109] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. "c-Through: part-time optics in data centers". In: *SIGCOMM Comput. Commun. Rev.* 40.4 (Aug. 2010), pp. 327–338. ISSN: 0146-4833. DOI: 10.1145/1851275.1851222. URL: https://doi.org/10.1145/1851275.1851222.

[110] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. "TopoOpt: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs". In: *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 739–767. ISBN: 978-1-939133-33-5. URL: https://www.usenix.org/conference/nsdi23/presentation/wang-weiyang.

[111] Philipp Wiesner, Ilja Behnke, Dominik Scheinert, Kordian Gontarska, and Lauritz Thamsen. "Let's Wait Awhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud". In: *Proceedings of the 22nd International Middleware Conference*. Middleware '21. Québec city, Canada: Association for Computing Machinery, 2021, pp. 260–272. ISBN: 9781450385343. DOI: 10.1145/3464298.3493399. URL: https://doi.org/10.1145/3464298.3493399.

[112] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. "Design, Implementation and Evaluation of Congestion Control for Multipath TCP". In: *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*. NSDI'11. Boston, MA: USENIX Association, 2011, pp. 99–112.

[113] Xin Wu and Xiaowei Yang. "DARD: Distributed Adaptive Routing for Datacenter Networks". In: *2012 IEEE 32nd International Conference on Distributed Computing Systems*. 2012, pp. 32–41. DOI: 10.1109/ICDCS.2012.69.

[114] Zhe Wu, Michael Butkiewicz, Dorian Perkins, Ethan Katz-Bassett, and Harsha V. Madhyastha. "SPANStore: Cost-Effective Geo-Replicated Storage Spanning Multiple Cloud Services". In: *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. SOSP '13. Farminton, Pennsylvania: Association for Computing Machinery, 2013, pp. 292–308. ISBN: 9781450323888. DOI: 10.1145/2517349.2522730. URL: https://doi.org/10.1145/2517349.2522730.

[115] Bahador Yeganeh, Ramakrishnan Durairajan, Reza Rejaie, and Walter Willinger. "A first comparative characterization of multi-cloud connectivity in today's internet". In: *Passive and Active Measurement: 21st International Conference, PAM 2020, Eugene, Oregon, USA, March 30–31, 2020, Proceedings 21*. Springer. 2020, pp. 193–210.

[116] Jin Y Yen. "Finding the k shortest loopless paths in a network". In: *management Science* 17.11 (1971), pp. 712–716.

[117] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing". In: *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX Association, Apr. 2012, pp. 15–28. ISBN: 978-931971-92-8. URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia.

[118] Sawsan El-Zahr, Paul Gunning, and Noa Zilberman. "Exploring the Benefits of Carbon-Aware Routing". In: *Proc. ACM Netw.* 1.CoNEXT3 (Nov. 2023). DOI: 10.1145/3629165. URL: https://doi.org/10.1145/3629165.

[119] Shawn Zandi. *Project Altair*. Mar. 2016. URL: https://engineering.linkedin.com/blog/2016/03/project-altair--the-evolution-of-linkedins-data-center-network.

[120] Mingyang Zhang, Radhika Niranjan Mysore, Sucha Supittayapornpong, and Ramesh Govindan. "Understanding Lifecycle Management Complexity of Datacenter Topologies". In: *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. Boston, MA: USENIX Association, Feb. 2019, pp. 235–254. ISBN: 978-1-931971-49-2. URL: https://www.usenix.org/conference/nsdi19/presentation/zhang.

[121] Yanwei Zhang, Yefu Wang, and Xiaorui Wang. "Greenware: Greening cloud-scale data centers to maximize the use of renewable energy". In: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer. 2011, pp. 143–164.

[122] Jiajia Zheng, Andrew A. Chien, and Sangwon Suh. "Mitigating Curtailment and Carbon Emissions through Load Migration between Data Centers". In: *Joule* 4.10 (2020), pp. 2208–2222. ISSN: 2542-4351. DOI: https://doi.org/10.1016/j.joule.2020.08.001. URL: https://www.sciencedirect.com/science/article/pii/S2542435120303470.

[123] Hua Zhong, PingPing Dong, WenSheng Tang, Bo Yang, and JingYun Xie. "A Short Flows Fast Transmission Algorithm Based on MPTCP Congestion Control". In: *Artificial Intelligence and Security*. Ed. by Xingming Sun, Jinwei Wang, and Elisa Bertino. Cham: Springer International Publishing, 2020, pp. 786–797. ISBN: 978-3-030-57884-8.

[124] Noa Zilberman, Gabi Bracha, and Golan Schzukin. "Stardust: Divide and Conquer in the Data Center Network". In: *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. Boston, MA: USENIX Association, Feb. 2019, pp. 141–160. ISBN: 978-1-931971-49-2. URL: https://www.usenix.org/conference/nsdi19/presentation/zilberman.

[125] Noa Zilberman, Eve M. Schooler, Uri Cummings, Rajit Manohar, Dawn Nafus, Robert Soulé, and Rick Taylor. "Toward Carbon-Aware Networking". In: *SIGENERGY Energy Inform. Rev.* 3.3 (Oct. 2023), pp. 15–20. DOI: 10.1145/3630614.3630618. URL: https://doi.org/10.1145/3630614.3630618.