

22. Evolving Requirements and Trends of HPC

Sébastien Rumley , Keren Bergman , M. Ashkan Seyedi, Marco Fiorentino 

High-performance computing (HPC) denotes the design, build or use of computing systems substantially larger than typical desktop or laptop computers, in order to solve problems that are unsolvable on these traditional machines. Today's largest high-performance computers, a.k.a. supercomputers, are all organized around several thousands of *compute nodes*, which are collectively leveraged to tackle heavy computational problems. This orchestrated operation is only possible if compute nodes are able to communicate among themselves with low latency and high bandwidth.

In 2004 the ASCI Purple supercomputer was the first to implement optical technologies in the interconnects that support these internode communications. However, research on optical interconnects for HPC applications dates back to the early 1990s. Historically, HPC has been a large driver for the development of short-distance optical links, such as the ones found in today's data-centers (as described elsewhere in this volume). As the number of research areas and industries that exploit HPC is growing, the need for improved HPC interconnection networks is expected to persist.

In this chapter we review the requirements of current HPC systems for optical communication networks and we forecast future requirements on the basis of discernible HPC trends.

22.1	Challenges of HPC	725
22.2	Defining High-Performance Computing	726
22.2.1	HPC and Datacenters	726
22.2.2	Quantitatively Measuring HPC	726
22.2.3	Economics of HPC	729
22.2.4	Parallel Nature of High-Performance Computing	730
22.2.5	End of Dennard Scaling and Beyond ...	731
22.3	Contemporary High-Performance Interconnection Networks	732
22.3.1	Interconnect Building Blocks	732
22.3.2	Injection Bandwidth	735
22.3.3	Topologies	736
22.3.4	Dragonfly Topology	741
22.3.5	Higher Networking Layers	744
22.3.6	Contemporary HPC Networks: Summary	745
22.4	Future of Optics in HPC Interconnects ..	746
22.4.1	VCSEL/MMF Link Technology	746
22.4.2	Silicon Photonics Technology	748
22.4.3	Other Forward-Looking Technologies for HPC Interconnects	750
22.5	Summary	751
	References	751

22.1 Challenges of HPC

High-performance computing (HPC) denotes the design and use of computers specifically engineered to tackle large computational challenges. A computational challenge comprises the correct execution of the instructions declared in a program and, possibly, the appropriate transformation of input data into output results. According to this definition numerous devices such as personal computers, laptops, and even smartphones or watches tackle a myriad of computational challenges every day. Traditional or embedded computers, however,

reach their limits when the challenge involves a very large number of operations, a massive amount of input, intermediate, or output data, or both. In those cases, traditional computers might at best deliver the correct output, but after an execution time so long that the results have very little or no value [22.1] (e.g., weather forecast delivered after the forecast epoch). In many cases, however, traditional computers will simply fail to execute the program altogether due to insufficient temporary storage space, to the presence of a random error in one of the

components, or to external causes such as a power failure. Some applications of computing thus require systems able to:

- Execute a large number of instructions per seconds to limit the total execution time
- Hold large enough intermediate states during the computation
- Be able to continue and/or resume correct operation for weeks or even months even in the face of a power or component failure.

Computers meeting these requirements are generally called high-performance computing systems.

Of course, requirements for a large number of operations per second, large memory and storage, and prolonged error-free operation are continuously growing as high-performance computing hardware evolves and progresses. Today's laptops and smartphones have capabilities that the most powerful supercomputers had about 20 years ago. High-performance computing can therefore be defined as the avant-garde of the computing industry. In particular, the field has seen the first

use of optical fibers directly within a computing system, and might prefigure a generalized use of photonics in general purpose computing (e.g., in interfaces for ultrahigh definition displays [22.2]), making this focus on HPC relevant in the context of this book.

In this chapter, the main characteristics and rationales of HPC, and of the associated high-performance interconnects, will first be reviewed. If HPC is the entry door to computing systems at large for the photonics community, it is worthwhile analyzing past, current and future requirements and trends of HPC in the context of this book. If these HPC requirements are not properly understood, the opportunity to enter the computing system market might be lost. The next section (Sect. 22.2) thus provides a digest of what a high-performance system is, and how it differs from other systems, e.g., from datacenters (Sect. 22.2.1). In Sect. 22.3, a description of HPC interconnection systems is provided. This description comes with more details as the interconnection network is the part with the highest need for optical technologies. Finally, in Sect. 22.4, we review the photonic technologies that might be deployed in next-generation HPC systems.

22.2 Defining High-Performance Computing

In the definition provided in the introduction of this chapter, HPC was compared to traditional computers, but HPC systems are also distinct from datacenters.

22.2.1 HPC and Datacenters

HPCs and datacenter *machines* have many things in common: they comprise a large number of compute cores, memories, and storage, consume large amounts of power, require heat dissipation and cooling, are operated 24/7, and often span large areas. Both execute a massive number of instructions per second. Datacenters and HPC systems can be mainly distinguished from each other by looking at the size of the tasks they execute. In an HPC machine, a *single task* can easily use most or even all resources and do so for days or even weeks. In datacenters, in contrast, most tasks are completed within seconds, and mobilize only a small fraction of the datacenter resources. A typical HPC task is a weather forecast simulation for a whole country, whereas a typical datacenter task consists of servicing a user web request. Consequently, datacenters currently use components and networks that are performing worse than their HPC counterparts. Requirements in terms of error and failure recovery, in particular, are less stringent in datacenters.

This situation, however, will change as the complexity of datacenter tasks grows. Computationally intensive tasks that would have been considered as high-performance computing a few years ago are emerging now in some datacenters. In-depth processing of video streams, for indexing or illegal content detection, is a typical example. Another example is natural language processing (NLP) [22.3]. A certain level of convergence between HPCs and datacenters is expected to take place in the next few years [22.4]. For this reason, HPC can be considered as the avant-garde for improved datacenter technologies as well, in particular for technologies that directly apply to hyperscale datacenters, as cooling, automated server management, and, obviously, interconnects.

22.2.2 Quantitatively Measuring HPC

High-performance computers come in a large variety of architectures, designs and sizes [22.5] suited for executing specific applications or application classes. Yet since the early days HPC practitioners have attempted to build a universal metric to compare HPC systems. Floating point operations per second (FLOPS) has become the gold standard for measuring performance of HPC systems. A floating point operation (abbreviated

flop) can be any arithmetic or logic one, e.g., +, −, *, /, AND, OR, SQRT, and so on, applied to operands of a specific precision. When operands have 64 bits, which is the default case in HPC, the FLOPS are said to be *full-precision*, while *single-* and *half-* precision FLOPS correspond to 32 and 16 bit operands respectively. Note that flop is a unit, and should be distinguished from the FLOPS unit, which represents flop/s.

FLOPS can represent either the number of operations a computing element is theoretically capable of delivering per second, in which case the term *installed* FLOPS is used, or the number of operations a computing element has effectively delivered while performing a computation (*delivered* FLOPS).

Counting Installed FLOPS

Determining the FLOPS *installed* in a system begins by identifying the FLOPS capability of each processing unit (PU) present in the system. A processing unit can be a CPU (central processing unit), a GPU (graphics processing unit), or any other digital logic component destined to perform computational operations.

To identify the FLOPS capability, one first has to consider the largest set of computational units that can be used at the same time, and determines how many *floating point operations* these units can perform in parallel. This number indicates the *width* of the PU and its unit is abbreviated *flop*. For example, a PU with four computational units each able to perform a fused multiply-add $d = ax + b$ can perform *width* = 8 *flop* simultaneously.

As a second step, the *width* must be multiplied by the *rate* to obtain the *flop/cycle* throughput of the PU. The *rate* indicates the number of cycles separating admissions of new batch of operations in the PU, and this regardless of the number of cycles required for this batch to be completed. In the ideal case, the *rate* is equal to 1, which means that a new batch consisting of *width* operations can be started at every cycle, and therefore that the PU micro-architecture is capable of carrying $width \times rate = width \times 1 = width \text{ flop/cycle}$. Note that a *rate* close or equal to 1 is rather common in modern micro-architectures.

The $width \times rate$ figure (in *flop/cycle*) must finally be multiplied by the frequency of the PU clock, expressed in *cycles per second* (thus, in Hz), to obtain the *installed* FLOPS capability of the PU (in *flop/s*). For example, if the PU with *width* = 8 *flop* that we considered above has a *rate* = 0.5 and is clocked at 3 GHz, its installed capability would be $8 \text{ flop} \times 0.5 \cdot 3 \times 10^9 \text{ s}^{-1} = 12 \times 10^9 \text{ flop/s} = 12 \text{ gigaFLOPS}$ (abbreviated GF).

The FLOPS installed in all the PUs in the system are finally summed up to yield the full-system installed capability. This generally works by multiply-

ing the number of PUs present within a chip (the term socket is also used) by the number of chips installed in every *compute node*, and finally by the number of compute nodes present in the system. The term compute node, often simply abbreviated as *node*, is widely used in the HPC community and can be assimilated to a server, in the sense that a node generally runs its own instance of the operating system and can work in relative autonomy. In contrast to conventional computing, however, compute nodes might not be individually boxed. The BlueGene architecture from IBM, for instance, consists of motherboards accepting 16 daughter boards, with two *nodes* on each daughter board [22.6]. Thirty-two nodes thus share a *blade*, which also provides a common power supply and cooling system.

At the end of 2017, Sunway TaihuLight was the publicly disclosed computer with the most installed FLOPS. Each CPU can execute 8 flop/cycle. With a rate of 1 and clock frequency of 1.45 GHz, each CPU delivers 11.6 GF. With slightly more than a million of such CPUs, the full system has an *installed* computing power in excess of 120 PF (petaFLOPS) [22.7].

Counting Delivered FLOPS

Installed FLOPS must be considered as a higher bound of performance, as they only reflect the maximum number of operations per second the system can perform. A system, however, rarely operates at full capacity, essentially for two types of reasons. First, the mix of operations to be performed at a given point in time may not fully match the capabilities of the computational units. In the above fused multiply example, the maximum width is reached when four $d = ax + b$ operations can be realized in parallel. Should the executed program only require multiplications, half of the PU capability would remain unused. Second, the input data required to perform the programmed operations might not be available yet, either because it must be fetched from memory, is en route from another PU, or is currently being calculated within a computational unit. All these circumstances will cause the PU to stall, most often for a couple of cycles but sometimes for much longer, while waiting for data to be available in the registers and causing a net loss of FLOPS compared to the installed capability. These leaks of FLOPS are not only unavoidable; they also reflect the quality of system design. Hence, it is not hard to realize that even the most powerful chip multiprocessor (CMP) will underperform without an adequate supply of data from memory and other CMPs. To judge the practical capability of an HPC system, the notion of *delivered* FLOPS is introduced.

The delivered FLOPS metric is obtained by dividing the number of flop required to complete a specific

calculation (generally a benchmark) by the time taken by the system to perform the calculations. The former number can be obtained either by annotating the code with operation counters and executing it, or by means of a complexity analysis of the underlying algorithm. One of the most famous of the benchmarks used to evaluate the delivered FLOPS, LINPACK (linear equations software package), requires the solution of a system of linear equations, which is known to require $2/3n^3 + 2n^2 + O(n)$ flop. The time taken to perform the calculations is generally experimentally measured, as predicting the number of PU cycles required, although theoretically possible, is generally intractable.

We note that the installed FLOPS metric is rather universal, possibly only subject to the interpretation of the width and rate parameters [22.8]. The delivered FLOPS metric of a given system, in contrast, depends on many aspects: the type of calculation performed by the benchmark application, the way it has been programmed, the compiler used and possibly even the input dataset. Different codes will challenge the components of an HPC system differently. The LINPACK benchmark, used in the *Top 500* ranking [22.9], is known to be extremely computationally intensive, stressing computational units primarily. Another benchmark called HPCG (high-performance conjugate gradient) [22.10], in contrast, stresses the communication infrastructure predominantly. Across multiple systems, the delivered FLOPS thus largely depends on the architecture and design choices. Two systems with equal installed FLOPS but different architectures generally deliver different delivered FLOPS when executing the same benchmark.

System Efficiency

The so-called system efficiency metric can be obtained by dividing the delivered FLOPS by the installed one. For the users of an HPC system, the efficiency denotes the intensity at which the tested benchmark leverages the computational resources. System designers, in contrast, will see the system efficiency as the result of efforts aimed at adequately connecting the computational units with the memory and among themselves to limit the number and duration of PU stalls. LINPACK calculations are relatively straightforward to conduct, even on highly parallel architectures (Sect. 22.2.4), and therefore system efficiencies for LINPACK are generally above 50% and sometimes above 90% [22.9, 11]. For HPCG, in contrast, efficiency rarely exceeds 10% and sometimes even falls below 1% [22.10].

Other Performance Evaluation Strategies

FLOPS-based performance evaluation methodologies are frequently criticized because they assign an over-

sized importance to computational units. As evidenced by the HPCG benchmark, computational units are not necessarily the performance bottleneck and, for some application, focus should be placed on other characteristics of the system. The Graph 500 is known as a benchmark that is not based on FLOPS [22.12, 13]. It comprises the pseudorandom construction of a massive graph with up to trillions of edges followed by a breadth-first search (BFS) over the graph. The performance is reported in terms of traversed edges per second (TEPS). This benchmark mimics calculations used in data analytics. The HPC system exhibiting the best performance on this benchmark, as of November 2017, is the six-year-old K computer, with 38 teraTEPS. This is more than the 24 teraTEPS demonstrated by Sunway TaihuLight (ranked second), even though the latter has more than 11 times more installed FLOPS.

Quantitatively Describing HPC Systems: Conclusions

Table 22.1 and Fig. 22.1 summarize the characteristics of some of the most powerful HPC systems active at the time of writing. FLOPS, installed or delivered, is a simple metric permitting description of the capabilities of an HPC system. As shown through the Graph 500 example, however, FLOPS counts are far from being a universal metric. Nevertheless, as we will see in the next section, FLOPS counts remain a powerful tool to scale various requirements, such as the memory bandwidth or communication bandwidth. It is therefore important to be aware of the FLOPS characteristics of the main systems.

In general, collocating many computational units within a single PU, and many PUs within a chip, requires engineering and technological exploits. Such exploits will boost a system's installed FLOPS, but offer no guarantee of increasing the delivered FLOPS and more generally improving the overall performance. In the design phase of a supercomputer architecture, HPC system designers face the choice of investing resources in installed FLOPS exclusively, which can cause a drop in the efficiency, or in other resources such as memory or the interconnection network that are likely to increase the overall efficiency. A system can be considered balanced for a given application when an additional dollar spent on installed FLOPS raises the delivered FLOPS by the same amount as a dollar invested on other components. For example, if adding more PUs in a system results in more delivered FLOPS, but boosting memory or interconnects of that system keeps performance unchanged, the computing part of that system is likely to be undersized compared to the noncomputing part.

Table 22.1 Main benchmarks for supercomputers

System name	Year of introduction	Top 500 rank	HPCG rank	Graph 500 rank	Delivered petaFLOPS (LINPACK/Top 500)	Delivered petaFLOPS (HPCG)	TEPS (Graph 500)
Sunway TaihuLight	2016	1	5	2	93 015	0.481	23 756
Tianhe-2 (MilkyWay-2)	2013	2	2	7 ^a	33 863	0.580	2061
Piz Daint	2016	3	4		19 590	0.486	
Gyokou	2017	4			19 135		
Titan	2012	5	9	3	17 590	0.322	
Sequoia	2012	6	8		17 173	0.330	23 751
Trinity	2017	7	3		14 137	0.546	
Cori	2016	8	7	1	14 015	0.355	
Oakforest-PACS	2016	9	6	4	13 555	0.385	
K computer	2011	10	1		10 510	0.603	38 621
Mira	2012	11	12		8 587	0.167	14 982
TSUBAME3.0	2017	13	10		8 125	0.189	
JUQUEEN	2013	22	19	5	5 009	0.096	5 848

^a Performance of a fractions of MIRA (ranked sixth, seventh and ninth in the graph500) excluded; if taken into account, Tianhe-2 ranks tenth

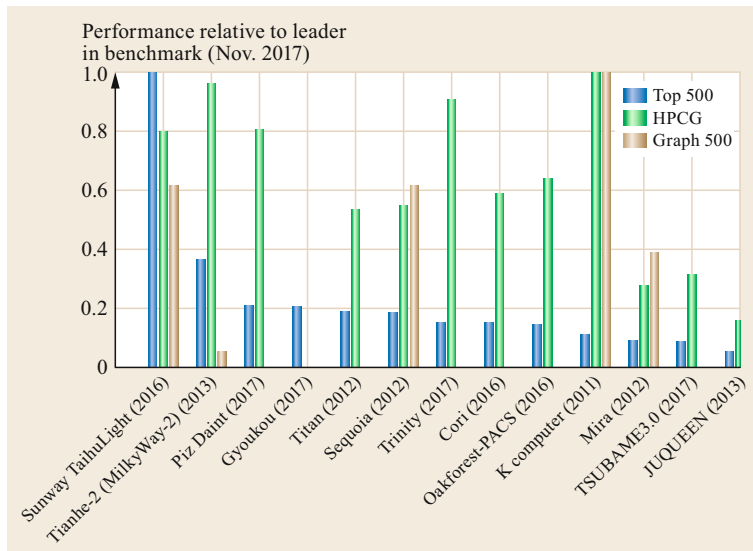


Fig. 22.1 Performance relative to leader of various supercomputers executing the Top 500 (LINPACK), HPCG (high-performance conjugate gradients) and graph500 (graph construction and search) benchmarks. Results from November 2017

22.2.3 Economics of HPC

Hyperion Research, a consultancy firm, estimated the HPC market to be worth \$ 11.2B in 2016, and segmented the market into four categories: i) the largest systems, priced over \$ 500k (and up to hundreds of \$M), denoted as supercomputers, ii) divisional systems priced between \$ 250k and \$ 500k, iii) departmental systems (\$ 100k–250k) and iv) workgroup systems (under \$ 100k) [22.14]. These categories represent respectively 37, 20, 28 and 15% of the HPC market. For comparison, IDC, another consultancy firm, estimates

the whole server market to be worth around \$ 60B. HPC equipment thus represents an important but not dominant share of the total server computer market (which does not include desktop or laptop computers, but does include datacenters).

As of November 2017, supercomputers, the above listed category of most interest for optical networking, were defined as systems with 200 TF (teraFLOPS) or more of installed computing power, and 6700 TF were sufficient to be listed in the 500 most powerful supercomputers. The number of supercomputers installed in 2016 is of the order of one thousand. Due to the scale

of the systems sold, the HPC market, and more particularly the supercomputer market, represent significant revenues and margins, but not many units sold.

The cost per installed and fully connected TF, which has been constantly falling thanks to progresses in very large scale integration (VLSI) and semiconductor fabrication processes, in 2017 were roughly on the order of \$2000. In 2017, a budget of \approx \$200M thus affords a 100 PF system. This amount of \approx \$200M has been frequently invoked by the US Department of Energy, one of the main operators of supercomputers, as the upper limit for the largest scale machines. To reach an EF (exaFLOPS) installed capacity, the next grand challenge in supercomputing [22.11], while remaining within a \$200M budget, the cost per TF should be reduced by an order of magnitude at least [22.15], and be brought below \$200 per TF.

Energy consumption costs and other operational costs (OpEx), if generally inferior to acquisition costs (CapEx), represent a significant portion of the total cost of ownership (TCO). By dividing the delivered FLOPS metric by the average power consumption, the energy efficiency of a system (expressed in FLOPS per Watt, or flop per Joule) can be obtained. The current rule-of-thumb is 1 GF/W (gigaFLOPS per Watt); as of November 2017, out of the 306 systems disclosing energy efficiency in the Top 500 benchmark, 204 were showing an energy efficiency metric better than this mark. Seven systems were also capable of delivering more than 10 GF/W. An energy efficiency of 6 GF/W has been measured when executing LINPACK on the *Top 500* benchmark leader Sunway TaihuLight [22.9].

Although relatively energy efficient, the Sunway TaihuLight system consumed an average of 15 MW during the execution. Such high power consumption, comparable to that of a town, requires special access to the electric grid, as well as special-purpose power transform stations, which increases direct and indirect costs. To avoid excessive power-supply overhead costs, it is commonly agreed that the power consumption of a supercomputer should be kept below 100 MW. The US Department of Energy, both one of the largest supercomputer operators, and the operator of some of the largest systems, discourages power consumptions beyond 20 MW. Reaching an exaFLOPS capacity with a 20 MW power budget requires energy efficiency to increase to 50 GF/W at least.

Supercomputers usually have lifetimes of five to seven years. For instance, the K computer became operational in 2011 and is still active six years later. As performance growth tends to decelerate (as it will be addressed in Sect. 22.2.5), longer lifetimes might become the rule. Thus, it is worth remarking here that operating a 20 MW-system for ten years yields a lifetime

consumption of 1752 MWh. At \approx \$75 per MWh, a fair estimate of the price paid by wholesale electricity consumers [22.16], the electricity cost amounts to \$131M, a figure not very far from to the \$200M acquisition cost mentioned above. This comparison shows that reducing power consumption can be relevant just from an operating cost perspective. Should the CapEx cost per TF be improved by a factor of ten in the next few years while the OpEx power per gigaFLOPS stays unchanged, an exaFLOPS-capable computer would consume 100 MW and thus cost \$655M over ten years. This 3.2 \times more than its acquisition cost, not taking into account extra power-supply station costs. Progress on the energy efficiency front must thus be realized alongside improvements in the cost efficiency [22.15].

22.2.4 Parallel Nature of High-Performance Computing

HPC has traditionally relied on parallelism to increase the advantage over mainstream computers while relying on the same semiconductor process for chip manufacturing. In fact, historically, most of the advances in processor architecture have been introduced in HPC systems first. The CDC 6600 system, introduced in 1964 and considered to be the first supercomputer, with its central processor equipped with ten functional units, was capable of working on different instructions at the same time [22.17]. This superscalar design, corresponding to a width factor larger than 1, is now prevalent in most modern processors. The CDC 6600 successor, the CDC 7600, introduced pipelining [22.18], i.e., the capability for a functional unit to work on several instructions at the same time, and thus to approach the aforementioned ideal *rate* of 1 (even though completing each instruction would take multiple cycles). This CDC 7600 was to be outperformed by the Cray-1 system in the early 1980s [22.19]. The Cray-1 supercomputer relied on vector processing, a feature present today in not only graphical processing units (GPUs), but also in conventional processors (Intel MMX, PowerPC AltiVec) [22.20]. Vector processing allows one instruction to be applied to multiple data at the same time, leading to even higher *width* values.

Besides introducing more parallelism within the PU, HPC has also relied on *spatial* chip- or node-parallelism to develop more powerful systems. Spatial parallelism means collocating multiple computing elements, usually all identical, within the same system, along with an interconnection network. The Cray X-MP supercomputer, which succeeded to the Cray-1 as the most powerful supercomputer in 1985, initiated the trend with two and then four interconnected vector processors. Other companies, however, quickly realized

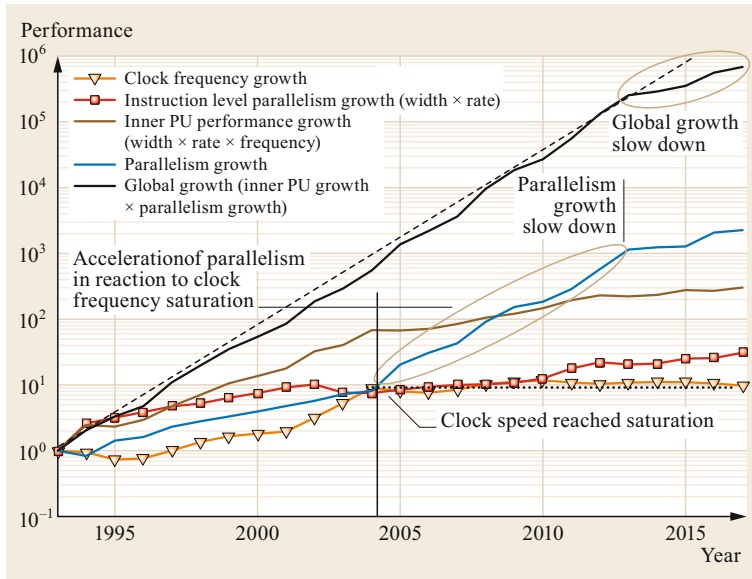


Fig. 22.2 Evolution of various contributors to overall top 20 HPC systems performance growth, relative to 1992. The *brown line*, representing the processing unit throughput, results from the product of the *orange triangle line* (processor frequency) and of the *red square line* (instruction-level parallelism). The *black line* results from the product of the *brown and blue lines* (processor parallelism). (Source: top500.org)

that combining many cheap components around an efficient interconnect could be more cost-effective than tailoring custom chips whose complexity was close to unmanageable. In the 1990s systems with more than one hundred processors came on the market. Fujitsu's Numerical Wind Tunnel, the fastest supercomputer in 1993, incorporated 140 processors [22.21], and Thinking Machines CM-5/1024, the second-fastest machine in 1993 according to the *Top 500* ranking, had more than one thousand processors.

This was only the beginning of establishing chip parallelism as one of the main drivers of HPC performance. As shown in Fig. 22.2, the number of cores present in the 20 most powerful supercomputers steadily increased during the 1990s.

22.2.5 End of Dennard Scaling and Beyond

As apparent in Fig. 22.2, from the early 1990s to the mid-2000s, the progression in chip- or node-level parallelism was still accompanied by a steady increase in each single PU capability, through increased clock speeds and PU inner parallelism (called instruction-level parallelism ILP). Progress in the miniaturization of transistors have driven this PU performance progression. *Dennard et al.* [22.22] stated in 1974 that by decreasing the size of a transistor by a factor k , the transistor operates k times faster, while its power consumption (at the faster rate) is divided by $\approx k^2$. This implies that power density is constant, i.e., a chip of surface S consumes roughly the same power, no matter if it carries thousands, millions or billions of transistors. Moreover, the more transistors, the faster the chip oper-

ating frequency. According to what has become known as Dennard's law, each new generation of semiconductor fabrication technology thus provided an additional budget of transistors that can be used to boost the PU performance, for instance through enlarged widths or rates closer to 1, along with an intrinsic speed-up through the clock frequency. It is worth noting that Dennard's scaling provides the physical grounds for the better-known Moore's Law. Moore's Law, in its initial version of 1965, postulated that the number of transistors in densely integrated circuits doubles about every 12 months. The projected rate of growth was later revised to every 24 months in 1975.

Dennard's scaling neglects a couple of phenomena, in particular leakage current and threshold voltage, deemed marginal in 1974 when Dennard's law was minted. After three decades of fast-paced scaling, however, these phenomena started to play a role after the turn of the millennium, and currently almost dominate the transistor power consumption. These effects partly caused the inner PU performance to plateau in the mid-2000s (as highlighted in Fig. 22.2). Hence, from thereon, PU clock frequency has been curbed to compensate for the extra power consumption due to leakage current and threshold voltage. The other reason for PU performance to saturate is that it has become more advantageous to collocate two or more identical CPUs on the same chip, something called core-level parallelism, than to further augment the complexity of a single PU through increased width and improved rates.

Core-level parallelism is nowadays the main performance driver, but it might not remain such for long. Moore's law is nearing its end [22.23]: leading semi-

conductor manufacturers currently operate (in 2018) at a resolution of ≈ 10 nm, leading to ≈ 60 nm-wide transistors. Industrial fabrication platforms allowing sub-10-nm resolution are in preparation, and IBM recently presented a 5 nm prototype process [22.24]. But resolutions below 2 nm are generally considered nearly impossible to achieve: that would mean structures with only 10 atoms (an atom of silicon, measured through its covalent radius, is 220 fm large). This means that the number of transistors available on a single chip will eventually be limited to around 100 billion, a factor ten from today. This portends ultimate chip-level FLOPS

performance of 50–100 TFs, as opposed to ≈ 5 TFs today.

Beyond this ultimate integration point, scheduled to happen around 2024 [22.25], further performance enhancements will almost exclusively be obtained by means of chip parallelism. In order to maintain the cost efficiency, chip designs will have to be simplified. Supercomputer housing and packaging (racks, power supply, etc.) will similarly have to be rationalized. In addition, and central to this chapter, interconnection networks will have to be tremendously improved in terms of cost and energy consumption.

22.3 Contemporary High-Performance Interconnection Networks

As shown in the previous section, installed computing power in excess of 5 TFs requires more than one chip, and requires these chips to be interconnected. For systems up to 50 TFs, it is possible to collocate up to a dozen of chips inside a single compute node (e.g., the recently introduced Nvidia DGX-1 station [22.26]), and to interconnect them using ad hoc links hardwired onto the motherboard. To reach higher installed computing power, however, there is little other option but to use multiple nodes side by side, and to introduce a high-performance interconnect to *glue* these nodes together in order to obtain a standalone system. An interconnection network is therefore a necessary and key component of any contemporary supercomputer. The goal of this section is to provide a comprehensive overview of the topology and characteristics of HPC interconnects. A correct understanding of these requirements is deemed necessary to correctly seize the role played by optical networking in HPC systems.

22.3.1 Interconnect Building Blocks

Figure 22.3 provides a high-level picture of an HPC interconnect, limiting the components to three types: adapters, routers and links.

Adapters

Adapters, also sometimes called NICs (network interface controllers), are the entry points to the network. The role of the adapter is obviously to inject packets into the interconnect access links with the appropriate format, as well as to collect incoming packets. High-performance NICs, however, also integrate advanced functionalities such as communication offloading [22.27], allowing a compute node to receive data from the network and store it into its memory while per-

forming other computations (unrelated to the transmitted data). This capability to compute and communicate in parallel keeps the system efficiency high, as it prevents installed FLOPS from being *wasted*. Adapters are often connected to the computing chip(s) using PCIe (peripheral component interconnect express) links. In some case, the adapter is integrated on the same die along with computational units (e.g., in the BlueGene architecture from IBM [22.6, 28]).

Discrete adapters pluggable into PCIe ports are being proposed by several vendors, for a price currently spanning from $\approx \$100$ to $\approx \$1000$, depending on the data rate and format. In 2017, for 100 Gb/s speeds, mostly relevant for modern interconnects, adapter price starts at \$450.

Routers

The *packet routers*, also denoted as *switches*, receive packets onto their N input ports and ensure these packets are forwarded with minimal delay to the correct N output ports. To minimize delays, a *cut-through* approach is adopted, i.e., the first bits of a packet are emitted onto the output port even though the last bits of the packet have not reached the router yet. High-performance routers are generally capable of extremely fast reaction times and, in the absence of traffic congestion, delay packets by tens of nanoseconds only [22.29]. Routers are also generally capable of delivering close to 100% throughput, i.e., they receive and send data on every link at maximal rate, as long as packets are adequately distributed onto output links. In addition to basic forwarding, routers are able to detect errors and apply quality-of-service (QoS) policies. QoS policing allows, for instance, traffic exchanged by computing resources, generally more sensitively to latency, to be processed with higher priority than traffic carrying output results to be stored onto the file system.

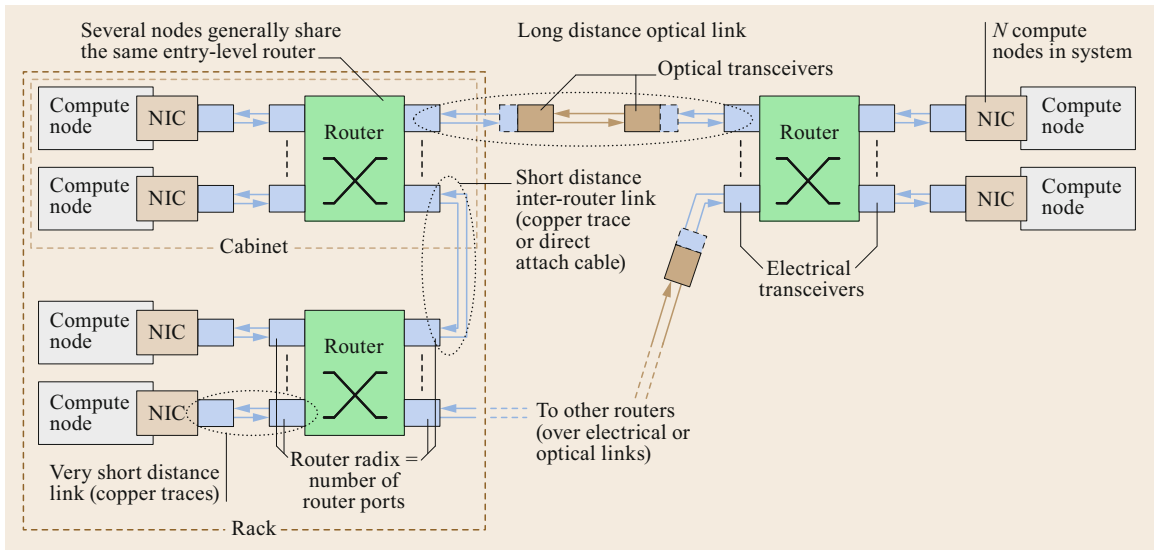


Fig. 22.3 High-level representation of the interconnect of a 2018 large-scale HPC system. Note that optical transceivers are used exclusively to connect routers not located within the same rack

From the HPC system architect perspective, routers are mainly characterized by the link-level and network-level protocols they implement, the data rate of the port, and the number of ports they provide (called the *radix*). The product of these last two figures gives the router *total switching bandwidth*, which represents the data amount that traverses the router and undergoes a switching operation every second. For example, a 36-port router operating at 100 Gb/s offers a 3.6 Tb/s total switching bandwidth. Note that router vendors sometimes present their product in terms of total I/O bandwidth, which is the sum of both ingoing and outgoing bandwidths, and thus equates to double the total switching bandwidth. In the above example, the router has a *total I/O bandwidth* of 7.2 Tb/s.

Modern HPC packet routers integrated on a single chip [22.29] provide up to 80 ports and link data rates up to 100 Gb/s, leading to total bandwidths reaching 5–10 Tb/s today (2017). They generally dissipate 100–200 W, leading to an energy dissipated per bit at full load of 10 to 40 pJ/bit. The router architecture reported by Scott et al. [22.29], which has been extensively leveraged in the recent systems from Cray [22.30, 31], a supercomputer vendor, shows a zero-load latency of 31.25 ns. This zero-load latency is the minimal amount of delay experienced by the first data bit of a packet transiting through the router.

Several vendors offer standalone boxed HPC routers including the router chip, a controller, fans, power supply, and connectors, for a price per port ranging from \$100 to \$500 per port and a price per Gb/s of \$1.5 to \$10, depending on the data rate and

number of ports. Routers delivering 100 Gb/s with 48 ports currently deliver the best cost/bandwidth ratio of ≈ 1.8 \$/Gb/s translating into \approx \$180 per port (\$8640 for the full equipment). A speed of 100 Gb/s was the highest offered by vendors, although products supporting 200 Gb/s were about to emerge. Products supporting 400 Gb/s are in preparation.

Multiple router chips can be combined inside a single package to offer an increased number of ports. The so-called resulting *director switches* can offer more than five hundred ports [22.32], at the expense of a significantly larger cost, higher per switched bit power consumption, and higher latency, reflecting the fact that most bits effectively traverse two or even three chips internally.

Links

HPC interconnect links connect adapters to their *entry-point router*, and routers among themselves. Links can be *active* or *passive*. In the latter case, the signals emitted by the adapter, the router chip, or a port-dedicated retiming chip are directly sent onto the transmission medium. This medium can be a copper-based cable, most often a coaxial cable with two inner conductors (called twinaxial, and often shortened twin-ax), or simply a pair of wires on a backplane. Differential signaling [22.33] is generally employed.

Active links, in contrast, have a transceiver on each end. A transceiver prepares and transforms the data received from the adapter or router for a transmission over a (generally) longer distance than passive links. It similarly receives the data after its transmission along

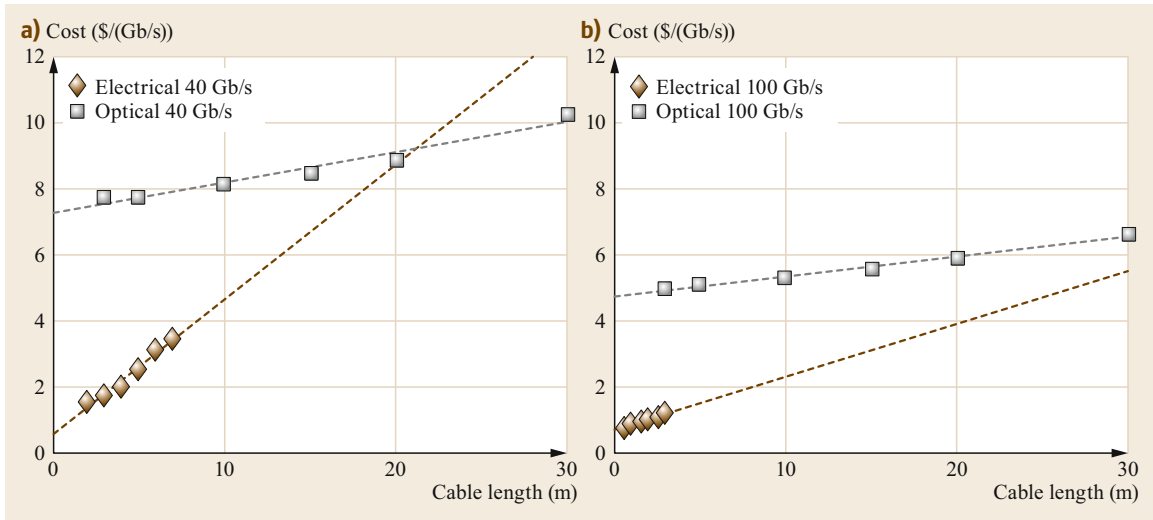


Fig. 22.4a,b Evolution of HPC links cost from 2014 (a) to 2017 (b). Clearly apparent is the reach limitation of electrical links, leading to a cost discontinuity (in 2017, from ≈ 1 \$/(Gb/s) for 4 m to ≈ 5 \$/(Gb/s) for 5 m distance)

the link and adapts it back to the requirements of the router or adapter port. Active links can be electrical, in which case the transceiver simply retimes and/or amplifies the signal prior to launching it onto the transmission medium. Most active links, however, are based on optical communication technologies. An optical active link is either composed of a pair of transceivers and a fiber cable of arbitrary length (within the range supported by the transceiver), or prepackaged with a fixed-length fiber. The latter option is called an *active optical cable* (AOC).

Links, active or passive, are mainly distinguished by their length, price and power consumption. Figure 22.4, which partly reuses data from *Besta* and *Hoefler* [22.34] illustrates the relationship between link lengths and costs, considering the cheapest commercial on-the-shelf links available at two points in time (2014 and 2017). Electrical links are roughly five times less expensive than their optical counterparts, in \$ per link terms, but are clearly limited in length. In 2014, the most economical data rate was 40 Gb/s, a speed that passive cables could carry over distances up to 7 m. In 2017, 100 Gb/s turned out to be the most economical data rate, but lengths of electrical cables were limited to 5 m. It is worth noting, however, that prices given in Fig. 22.4 correspond to discrete components, to be bought separately and individually. Cheaper electrical links can be obtained by integrating many copper traces onto a single motherboard, for instance, as well as by ordering large quantities.

The transmission technology over metallic wires causes electrical links to abruptly turn uneconomical beyond a certain point. This overall results in a price

discontinuity, clearly visible in Fig. 22.4, between short connections (currently below 5 m) and longer ones. As of today, and as shown in Fig. 22.3, metallic wires are used to connect chips among themselves on the same motherboard (chip-to-chip links), motherboards among themselves within a *chassis* or *cabinet* (intra-cabinet links), and to connect cabinets located within the same *rack* (intrarack links). Several neighboring racks may even be interconnected with metallic wires. Optical links, in return, are used for interrack links exclusively. As will be seen in Sect. 22.3.4 describing the Dragonfly topology, the major price difference between electrical and optical links have so far encouraged interconnect designers to develop interconnection schemes that make the most use of electrical cable and limit the number of optical connections required.

It is important to note that electrical cables can be expected to become uneconomical for shorter distances as data rates evolve toward 400 Gb/s and beyond (as summarized in Sect. 22.3.6). Optical links, in contrast, are expected to become more economical thanks to higher integration, as will be evoked in Sect. 22.4. Overall, one can expect optics to eventually equip all intercabinet links (i.e., roughly longer than one meter), and possibly some intracabinet ones (> 30 cm).

In terms of power consumption, the most energy efficient active optical links consume 15–25 mW/(Gb/s) (or equivalently pJ/bit, as Watt is equivalent to J/s). A 100 Gb/s link thus consumes about 4 W, i.e., 2 W per extremity. Such an energy efficiency figure, of the order of 10 pJ/bit, might seem a lot provided that numbers in the 1 pJ/bit range or even below are frequently evoked in the literature (e.g., as summarized in the survey from

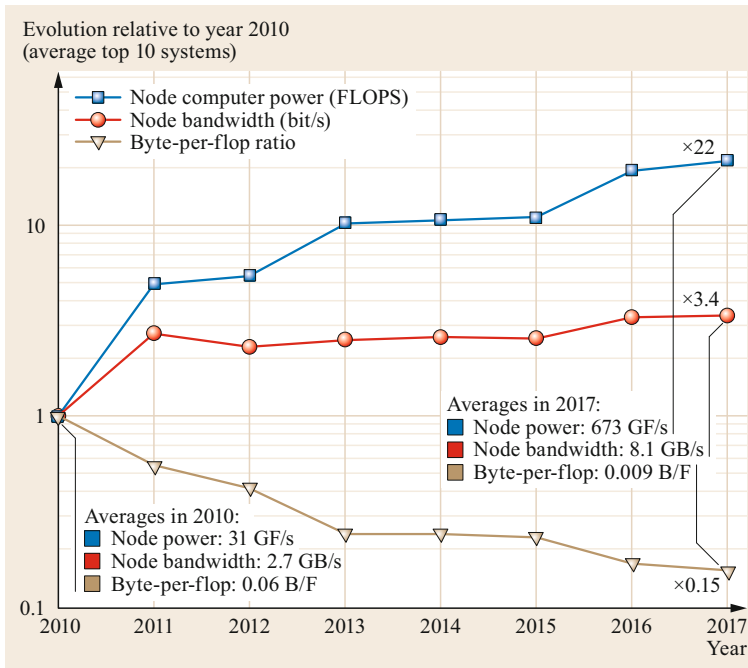


Fig. 22.5 Relative evolution of the node computer power, of the node injection bandwidth, and of their ratio over the years, for the top 10 systems. If the average node compute power has increased by a factor of 22, the injection bandwidth has risen by a factor of 3.4 only, leading to a 6.5 \times decrease of the byte-per-flop ratio

Thraskias et al. [22.35]). However, it is important to note that this power consumption corresponds to the amount of power that is driven from the host, and thus includes everything from the laser, through the modulator, the driver, to the serializer, and to the electrical interface. The power consumption of the latter is seldom included in the literature, in which articles often focus on the transmission subsystems, and not on the full system.

The energy dissipation of the active optical cable adds to the energy dissipated by the router and adapter to send, receive, and transmit data. The latter is generally limited to 2–3 pJ/bit [22.36].

22.3.2 Injection Bandwidth

The high-level representation of an HPC interconnect shown in Fig. 22.3 does not illustrate two very important aspects of the design of an HPC interconnect: the bandwidth of the system links, and the way routers are interconnected, i.e., its topology. We start by describing the former.

The amount of bandwidth a compute node has available to *reach out* to the other nodes through the interconnect is called the *injection bandwidth*. The injection bandwidth is generally set in proportion of the FLOPS installed on the node. A scaling factor in *byte/s/FLOPS* is thus introduced to obtain the byte/s required by a node with a given amount of installed FLOPS. Note that the byte/s/FLOPS unit is gener-

ally simplified in *byte/flop*, where flop stands again for floating point operation, a terminology that is also more intuitive. Note also that for convenience, 1 byte/flop is often approximated with 10 *bit/flop*. Fundamentally, 1 byte = 8 bits but due to transmission system overheads as parity checks or line codes, more than 8 bits are *physically* sent over a link for each byte. The 1 byte \approx 10 bits approximation accounts for these overheads.

The injection bandwidth scaling factor of an existing system indicates how many bytes can be communicated onto the interconnect for each executed flop before reaching network congestion, which itself causes execution slowdown. Injection bandwidth scaling factors of leading supercomputers are between 0.01 and 0.001 byte/flop and show a decreasing trend over time, as shown in Fig. 22.5.

The injection bandwidth scaling factor is strongly related to the *verbosity* factor, that denotes how many bytes are *listened to* and *spoken out* each time a flop is executed. *Verbosity* factors can be defined for various components of the supercomputer node. At the very core of the PU, the *verbosity* is of 24 byte/flop at least, as an operation requires two 8-byte (64 bits) operands, produces an 8-byte result and requires an instruction. However, operands or results are often almost immediately reused for another operation, and thus do not need to be read again from or stored in the memory. Through reuse, the *verbosity* of the main memory can diminish to \approx 0.1 bytes/flop (a figure generally considered by HPC node designers) or even less. The ability to reuse

operands, however, is highly application-dependent. The aforementioned HPCG benchmark, for instance, has a *memory verbosity* of 4 bytes/flop [22.37], which alone almost explains why system efficiencies while executing the HPCG benchmark seldom exceed 10% (Table 22.1). The *network verbosity* factor is similarly dependent on the application, but furthermore depends on the size of the problem being solved, the number of nodes solving it, and the way the application is mapped to these nodes [22.38]. As such, it is hard to predict, but can be measured experimentally [22.39]. In general, network verbosity factors tend to decrease as computational power of nodes is increased: with larger nodes, a larger proportion of the data exchanges between PUs remain concealed within the nodes, and does not emerge onto the interconnect. This effect, known as the surface-to-volume ratio, partly explains the decreasing injection bandwidth scaling factor trend shown in Fig. 22.5.

Historically, nodes have been most often equipped with a single adapter to provide the node's entire injection bandwidth. However, the recent increase in the number of installed FLOPS per node, which mainly drove the computing power progression in the last few years (Fig. 22.5), calls for a multiplication of adapters. Typically, a 40 TF node such as the Nvidia DGX-1 station needs 40 GB/s of bandwidth just to support a network verbosity factor of 0.001 byte/flop. As no single adapter on the market can deliver 40 GB/s of bandwidth yet, the station integrates four independent adapters each running at ≈ 10 GB/s. The term *double-rail* or *quad-rail* is sometimes used to denote node architecture with multiple adapters.

Obviously, multirail node architectures put larger demand on interconnect topology scalability (as will be defined in the next subsection). A thousand-node system with quad-rail adapters requires, for instance, an interconnect supporting 4000 end-points. An end-point is a generic term that describe a *leaf* connected to a topology.

22.3.3 Topologies

Next to the node injection bandwidth and associated supported *network verbosity* factor, the *topology* is the most notable characteristic of an HPC interconnect. The topology defines how end-points are connected to their entry-point routers, and how these routers are connected among themselves. Neglecting lost packets, which seldom occur in HPC, an interconnect can be seen as an abstract *packet delayer*: any packet injected onto the interconnect is eventually delivered to its destination node with some delay. The role of the topology essentially consists of minimizing these delays, given the available

hardware and injection bandwidth (dictated by a network verbosity factor).

To minimize delays, an HPC topology must first ensure that any node can reach any other node through a minimal number of hops over the network. This minimizes latency as each hop comes at the expense of some zero-load latency. To that end, the average distance of the topology should be minimized to guarantee low delays on average. However, the worst-case delay, which can be highly deleterious in parallel environments [22.40], should also be minimized. This is why topology architects pay very special attention to the topology *diameter*. The diameter is the number of hops in the longest of the minimal paths connecting one vertex to another, and thus a proxy for the worst-case zero-load latency. If the diameter of the topology is D , and realizing a hop in the topology takes about 50 ns, as reported for instance for the Blue Gene/Q interconnect [22.28], a packet will be delayed by $50D$ ns at least.

In addition to ensuring low *baseline*, zero-load latency due to propagation along links and traversal of routers, the topology must ensure that in most practical cases, as few packets as possible will experience too-long delays due to queuing. To achieve this, traffic injected should find sufficiently ample and lightly loaded paths onto the network to reach its destination.

By scaling the bandwidth available for a given topology (e.g., by doubling the bandwidth available on every link), one generally improves performances, so the overall bandwidth provided by an interconnect is an important quality factor. At the same time, two topologies comprising the same number of links and the same aggregated bandwidth (summed over all links) might yield distinct performances [22.41]. A good topology is a combination of an adequate quantity of routers and bandwidth with a deft recipe that describes how routers and end-points are connected. As for injection bandwidth, topology requirements depend on the application(s) execution in the supercomputer.

HPC topologies can be direct or indirect. In direct topologies, every router has end-points attached to it, whereas indirect topologies involve *inner* routers not directly exposed to the end-points. In the remainder of this subsection, the properties of some of the most common and exemplary topologies are detailed. Note that mathematical developments leading to scaling equations are omitted for the sake of conciseness.

Indirect Topologies

Fat-Trees. Indirect topologies are generally called fat-trees: end-points are the leaves of the tree, and traffic progresses up the root until it reaches a common ancestor with the destination, at which point it is routed downwards back to the destination end-point. The num-

ber of routers that must be traversed to reach the root of the tree (including the root router) indicates the number of levels present in the tree *minus one*. For instance, in the fat-tree constructs visible in Table 22.2 (to be defined next), three routers must be traversed to reach the top level, thus all these constructs are two-level fat-trees. The diameter of the fat-tree, excluding hops from and to end-points, is simply the double of the number of levels.

If the total bandwidth available between any consecutive levels of the tree is constant, and equal to the total bandwidth connecting the end-points to the entry-point routers, a fat-tree is said to be fully provisioned. As long as the traffic destined to one particular end-point does not exceed the end-point's ejection bandwidth, fully provisioned fat-trees are immune to congestion, even if all the traffic must progress up to the root of the tree. Fully provisioned and populated fat-trees with k levels and constructed with routers with radix r (thus offering r ports) have in total $r(r/2)^{k-1}$ ports available for end-points. If each compute node occupies a single end-point, the corresponding HPC system can scale up to $r(r/2)^{k-1}$ nodes. Fully provisioned fat-trees involve $(2k-1)(r/2)^{k-1}$ routers with $(k-1)r(r/2)^{k-1}$ internal links. An internal link is a link that connects two routers and thus that is not connected to an end-point. This corresponds to $k-1$ internal links per end-point, or k links in total (counting the link connecting the end-point to the interconnect). This also corresponds to $(2k-1)/r$ routers per end-point.

These expressions in *per end-point* terms are handy as they permit us to quickly evaluate the interconnect cost in proportion to the end-point's one. Consider, for example, a fully provisioned fat-tree with $k=2$ levels made of routers with $r=48$ ports, and accepting up to $r(r/2)^{k-1} = 1152$ end-points. With a router and link prices of \$8000 and \$80 (passive links, Fig. 22.4), corresponding to 100 Gb/s data rate, the *cost per end-point* is in this case $(2k-1)/r \times \$8000 + (k-1) \times \$80 = (3/48 \times \$8000) + (2 \times \$80) = \$660$. This cost per end-point must be compared to the cost per TF discussed in Sect. 22.2.3. If the budget for a 1 TF node (interconnect included) is \$2000 and if this node is connected through a single adapter, the interconnect cost represents about a third of the cost. We note that connecting a 1 TF node through a single 100 Gb/s adapter yields an injection bandwidth scaling factor of 0.001 bit/flop \approx 0.01 byte/flop. For a three-level fat-tree, scalable up to 27 648 end-points, the cost per end-point grows to $(\$8000 \times 5/48) + (3 \times \$80) = \$1073$, in which case interconnect equipment would represent more than a half of the budget. These example calculations illustrate how interconnect equipment cost prevents the injection bandwidth scaling factor from being kept relatively high.

Slimmed Fat-Trees. To mitigate the relatively high cost of fully provisioned fat-trees, oversubscription is often applied to fat-trees, which then become *slimmed fat-trees* or *tapered fat-trees* [22.38]. Oversubscription consists of allocating more ports downward the tree than upward. For example, out of the 48 ports of a router, 36 can be used to connect end-points, while only 12 are used to connect to the higher levels. In this case the oversubscription ratio is 36 : 12 thus 3 : 1. Applying an oversubscription of $x : 1$ allows us to inflate the scalability of the fat-tree by $2x/(x+1)$. The scalability of a two-level fat-tree with a 3 : 1 oversubscription ratio applied to either level thus grows by $2 \times 3/4 = 1.5$, while the scalability of a three-level fat-tree with two successive 2 : 1 oversubscription ratios is multiplied by 16/9, and thus increases by about 78%. Table 22.2 displays all possible ways to oversubscribe a two-levels fat-tree with $r=6$.

Oversubscription permits us to enlarge the fat-tree scalability for a given number of levels k and router radix r . This permits us to connect more nodes while conserving a low diameter (this will be further discussed in next subsection). Oversubscription also permits us to reduce the number of routers and links per end-point. Assuming that the scalability offered by the resulting fat-tree is fully utilized, i.e., that

$$\left(\frac{r}{2}\right) \frac{2x_1}{x_1+1} \left(\frac{r}{2}\right) \frac{2x_2}{x_2+1} \dots r,$$

end-points are interconnected. The number of routers required per end-point is reduced by a factor

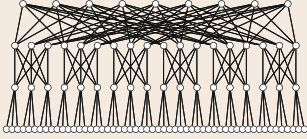
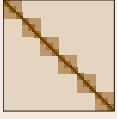
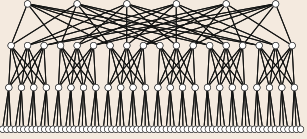
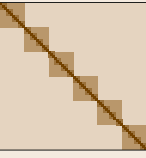
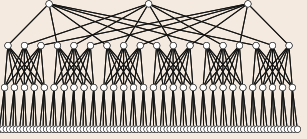
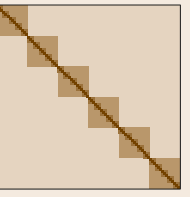
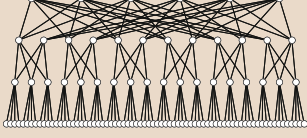
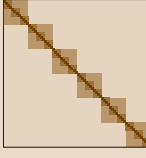
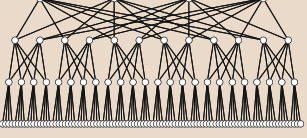
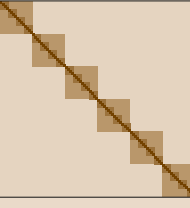
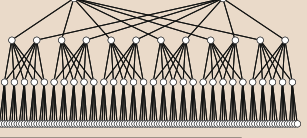
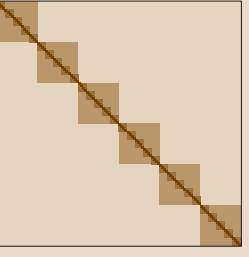
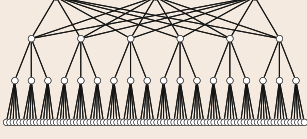
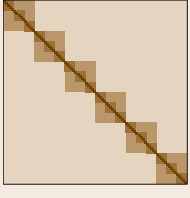
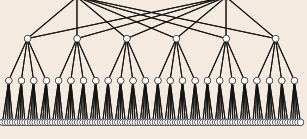
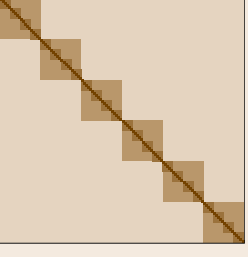
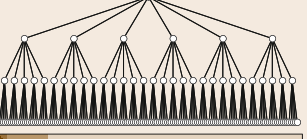
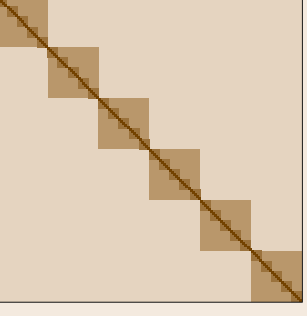
$$\frac{2k-1}{1 + \frac{2}{x_1} + \frac{2}{x_1x_2} + \dots}$$

and the number of internal links per end-point by a factor

$$\frac{k-1}{\frac{1}{x_1} + \frac{1}{x_1x_2} + \dots},$$

where x_i is the oversubscription realized between level i and $i+1$. A two times repeated 2 : 1 oversubscription thus permits us to reduce router costs by 50% and cabling by 62.5% in a three-level fat-tree. Considering a three-level fat-tree, $r=48$ and the aforementioned prices, this permits us to reduce the cost per end-point down to \$557 from \$1073 while extending the scalability to close to 50k end-points. Figure 22.6 and Table 22.2 aim at graphically illustrating the impact of oversubscription by considering the nine options available with radix $r=6$. Figure 22.6 shows how routers and link per end-point figures, and consequently cost

Table 22.2 Oversubscription options for two-level fat-trees with radix $r = 6$

	$x_2 = 1$	$x_2 = 2$	$x_2 = 5$
$x_1 = 1$	  Scalability = 54 Routers: 45 Internal links: 108	  Scalability = 72 Routers: 48 Internal links: 108	  Scalability = 90 Routers: 21 Internal links: 108
$x_1 = 2$	  Scalability = 72 Routers: 36 Internal links: 72	  Scalability = 96 Routers: 40 Internal links: 72	  Scalability = 120 Routers: 44 Internal links: 72
$x_1 = 5$	  Scalability = 90 Routers: 27 Internal links: 36	  Scalability = 120 Routers: 32 Internal links: 36	  Scalability = 150 Routers: 37 Internal links: 36

Hops separating end-point pairs: 0 2 4 6

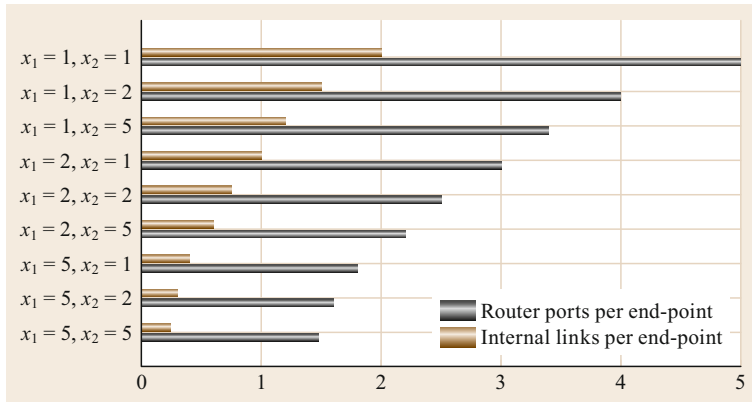


Fig. 22.6 Effect of oversubscription factors on a two-level fat-tree with six-ported routers. If no oversubscription is applied ($x_1 = x_2 = 1$), two internal links (one per level) are associated with each end-point. These two internal links occupy four router ports, in addition to the *entry* router port used by the end-point, thus five ports in total. On the contrary, if the highest level of oversubscription is applied at both levels ($x_1 = x_2 = 5$), the number of internal links per end-point falls drastically to ≈ 0.24 . This demonstrates that oversubscription is a powerful tool to tailor the performance, and thus cost, of an interconnect

per end-point, decrease as oversubscription factor x_1 and x_2 are increased. Table 22.2 shows how the scalability, and the routing distances, are affected.

Substantial savings can be obtained by means of oversubscribed fat-trees, but over-subscription also imposes limits on the verbosity supported in specific cases [22.38]. Consider once again an 2 : 1 oversubscription ratio applied twice. The maximum network verbosity factor supported is divided by a factor of four if the entire traffic needs to progress up to the tree root to reach the final destination. Oversubscription factors should therefore be determined with adequate knowledge of the expected traffic patterns. If intense traffic flows are mostly concealed along the diagonal of the traffic matrix, indicating that end-points mostly exchange traffic with the end-point with the next immediate index, relatively high oversubscription ratios can be tolerated. In those cases the traffic pattern is said to show locality. In contrast, if traffic is expected to be mostly exchanged between end-points separated by two or three fat-tree levels, and therefore has to operate four or six hops to reach its destination, oversubscription will act as a bottleneck.

It is worth nothing that when the largest part of the traffic must reach the root of an overprovisioned tree, i.e., for traffic patterns without locality, the bottleneck at the root level renders the injection bandwidths underutilized. This shows that for traffic patterns without locality, it only makes sense to linearly reduce the available bandwidth everywhere in the tree if costs have to be compressed. For traffic patterns with locality, in contrast, costs can be saved by gradually tapering bandwidth across the tree levels, following the effects of

locality. The effect of fat-tree network configurations on realistic workloads executed on very large systems has been analyzed in the literature, notably by *Jain et al.* [22.42]

Also note that more complex constructs of tree-inspired topologies exist, in particular the orthogonal fat-tree, which trades immunity to adversarial traffic patterns for improved scalability [22.43].

Directed Topologies

Fat-trees oblige traffic to take an even number of hops through the topology. Two hops are minimally required to ingress and egress the interconnect. However, unless two end-points share the same *entry-point router*, their communications must involve at least four hops, including hops from and to an end-point (see topology examples in Table 22.2).

It is possible to reduce this minimal number of hops for nonentry router-sharing end-points to three by considering *direct* topologies where every router acts as an entry point for end-points. The term direct topology originates from designs where compute nodes fulfill routing functionalities themselves, thus communicate *directly*. Historically, direct topologies also often tried to reproduce the locality patterns shown by specific distributed applications.

Before diving into descriptions of the most common direct topology constructs, it is worth analyzing the per end-point requirements in terms of links and routers of direct topologies.

Consider first that in the worst case, all traffic travels over as many hops as the diameter D . Under this assumption, each bit of traffic will leave a router D

times for another router, leave a router once for the final destination, and thus leave $D + 1$ routers in total. Over the r ports of a router, $r(D/(D + 1))$ ports should thus be assigned to links connecting routers, leaving $r(1/(D + 1))$ ports available on every router to accommodate end-points.

From there, one can deduce that the number of end-points per router is $r/(D + 1)$, and thus that the router per end-point is $(D + 1)/r$. For example, taking $D = 3$ and $r = 48$, $r(D/(D + 1)) = 48 \frac{3}{4} = 36$ ports out of the 48 will be devoted to internal links, while $r(1/(D + 1)) = 12$ ports are available on each router for end-points. The number of routers per end-point is $1/12$. Considering again a price of \$8000 for such a router, the *router cost per end-point* is \$666.

Further considering all routers similar among themselves, a router offers r ports and accommodates $r/(D + 1)$ end-points, so each end-point requires $r/(r/(D + 1)) = D + 1$ ports, and thus $(D + 1)/2$ links (as two ports necessarily translate into a link). As a general rule, diameter 2 and 3 direct topologies thus require ≈ 1.5 and ≈ 2 links per end-point respectively, including the link connecting the end-point to the interconnect, to offer enough bandwidth in total to carry traffic over the longest paths. Situations with fewer links per end-point denote oversubscribed direct topologies.

Toruses. A typical example of a topology mimicking a traffic pattern is the 3-D torus, that is ideally suited for so-called 3-D stencil-based parallel applications. This can be, for instance, a fluid dynamics simulation in a 3-D volume by means of finite-element methods. A portion of the 3-D volume (a cube) is assigned to each compute node. At each step of the simulation, the compute node must exchange boundary conditions with the six other nodes responsible for the six neighboring cubes. If these six nodes are directly connected to the original compute node, traffic undertakes a minimal number of hops onto the topology. IBM's BlueGene/L supercomputer series was based on a 3-D torus [22.6].

A 3-D torus requires two ports to navigate on each dimension with the six neighboring nodes, plus at least one port to connect to the end-point. A 3-D torus is thus advantageous for its low requirement in number of ports per router of seven, independent of the size of the interconnect. However, if very suited for 3-D stencils, 3-D toruses are badly suited for arbitrary traffic patterns. Hence, the worst-case number of hops required in a 3-D torus is proportional to the cube root of the number of end-points. A 3-D torus of size $11 \times 11 \times 11$ shows for instance a diameter of 15. This diameter far exceeds that of a three-level fat-tree with diameter of six only.

To offer better support to arbitrary traffic patterns, toruses can be constructed in higher dimensions.

For instance, the above-mentioned K computer uses a six-dimensional torus of dimension $2 \times 2 \times 3 \times 5 \times 5 \times 6$ [22.44]. IBM's BlueGene/P and Q series employed a 5-D-torus interconnect [22.28]. Diameters in 5-D or 6-D toruses are reduced to fifth- or sixth-degree roots, but nevertheless remain proportional to the scale. To alleviate this scaling limitation, direct topologies offering constant diameter (within their scalability limits) have also been proposed and are introduced next.

Flattened-Butterfly Topologies. As in toruses, routers in a flattened-butterfly topology [22.45] are organized as an n -dimensional lattice (where $n > 1$), but instead of connecting them with their neighboring routers exclusively, they are connected to all other routers that share at least one coordinate on the lattice. If the lattice is of dimension 4×4 , the router at coordinate $(0, 0)$ will be connected to $(0, 1)$, $(0, 2)$ and $(0, 3)$ along the x -axis, and $(1, 0)$, $(2, 0)$ and $(3, 0)$ along the y -axis. This construct, also called Hyper-X [22.46], has diameter n , as in the worst case one hop is required on each dimension. According to the above-mentioned method, $rn/(n + 1)$ ports among r can be devoted to other routers ($r/(n + 1)$ per dimension). The lattice can be thus of size $1 + r/(n + 1)$ in every dimension, leading to a number of routers of $(1 + r/(n + 1))^n$ and finally to a scalability of

$$\left(\frac{r}{n+1}\right) \left(1 + \frac{r}{n+1}\right)^n.$$

In the one-dimensional case, which corresponds to a full-mesh topology in which any router is connected to any other router, the diameter is one, so $r/2$ ports are allocated to end-points, and the $r/2$ remaining ports are used to connect to $r/2$ other routers. 1-D flattened butterflies thus involve $r/2 + 1$ routers in total, leading to scalability of $r/2(r/2 + 1)$. With $r = 48$, 600 end-points can be accommodated.

In the 2-D case, the topology is scalable to up to 4624 end-points with $r = 48$. This is vastly superior to the two-level fat-tree, which scales to 1152 end-points only, for an identical number of hops. However, as shown in Fig. 22.7, if n -dimensional flattened butterflies have overall all the bandwidth required to support as many two-hop flows as end-points, this bandwidth is not necessarily available at the right place and congestion might occur, something a fully provisioned fat-tree is immune against.

Moore Bound and Scalability Limits

Being able to use a topology of diameter $D - 1$ instead of D generally permits us to decrease global interconnect latencies by $D/(D - 1)$, and the number of links

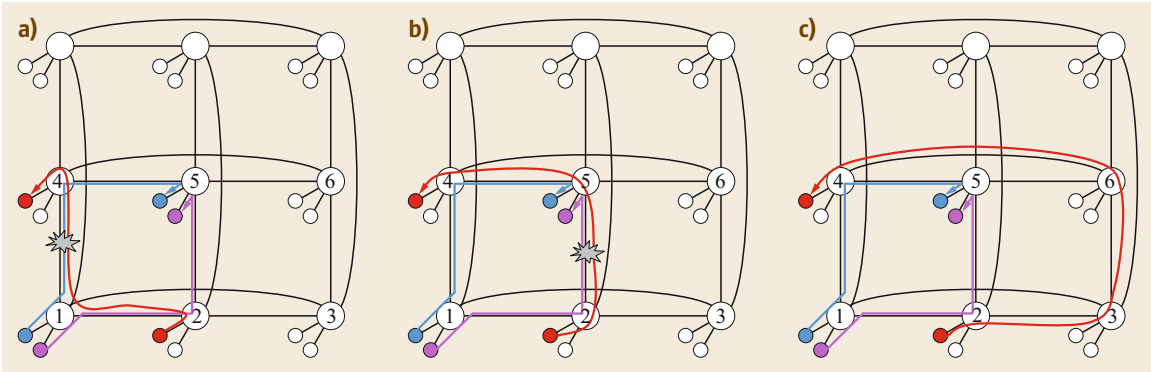


Fig. 22.7a–c Example case of blockingness in a 2-D-flattened butterfly even though only $r/(D+1) = 6/(2+1) = 2$ end-points are connected to each router. As long as two routers do not share a coordinate, there are D shortest paths available between them. The $D = 2$ shortest paths leading to router 4 from 1 are shown in *blue* and *magenta*. If the two end-points connected to router 1 need to send data at full rate to the two end-points connected to router 4, these two shortest paths can be utilized. However, this allows no bandwidth on the shortest paths leading from router 2 to router 3: neither on path 2–1–3 (a) nor on path 2–4–3 (b). Traffic exchanged by the end-points shown in *red* can be routed over a nonminimal distance path, for example over 2-3-6-4 (c). However, this path has three hops and thus consumes three units of bandwidth, one more than the $D = 2$ units allocated to each end-point

and routers per end-point by $(D+1)/D$. Specifically, achieving diameter 2 instead of 3 saves 33% in latency and 25% on link and router costs, while achieving diameter 3 instead of 4 decreases latency by 25%, and costs by 20%. There is therefore a great incentive to find low-diameter topologies with large scalability.

A straightforward way to decrease the diameter is to use routers with more ports. For instance, a three-dimensional flattened butterfly is required to support 15 000 end-points with 48 port routers. However, if routers with 72 ports are available, a 2-D flattened butterfly becomes possible (as a 25×25 lattice containing 625 routers, each supporting 24 end-points). Routers with higher numbers of ports, however, may not be available or may exhibit high cost/performance ratios, as developed in Sect. 22.3.1, *Routers*. There is therefore a strong incentive to construct topologies whose scalability is maximal for a given diameter and number of ports.

A scalability bound for topologies of diameter D built around routers with r ports can be obtained through the following steps. Assume again that $R = r(D/D+1)$ ports are allocated to connections between routers. Let us call this parameter R the internal radix, i.e., the radix of the interconnect facing ports. From a given *origin* router in the topology, $R = r(D/D+1)$ other routers are reachable in one hop; we call these *level 1* (L1). Through each of these R L1 routers, another $R-1$ L2 routers at most can be reached in two hops, thus no more than $R(R-1)$ routers in total can be reached in two hops. Following a recurrence relationship, no more than $R(R-1)^{D-1}$ routers can be reached in D hops. That generally means that a topol-

ogy of diameter D cannot have more than $1 + R + R(R-1) + \dots + R(R-1)^{n-1}$ routers. Notice that each router different than the root one has one port deducted from R to connect to its *parent* router. As each router can accommodate $r/(D+1) = R/D$ end-points, no more than

$$\frac{R}{D} [1 + R + R(R-1) + \dots + R(R-1)^{D-1}] \quad (22.1)$$

end-points can be supported. For $D = 2$ and $r = 48$, $R = 32$ and the maximum scalability is 16 400. This is more than three times the scalability of the 2-D flattened butterfly. This scalability bound, evaluated for different diameters and as function of radix in Fig. 22.8, is called the Moore bound (after Edward F. Moore, not to be confused with Gordon Moore of Moore's Law). *Besta* and *Hoefler* et al. [22.34] have shown that diameter 2 topologies achieving more than 80% of the maximal Moore bound scalability could be found, as well as diameter 3 topologies achieving more than 60% of the bound.

The Moore bound, as explained above and given by (22.1), considers that always $R(R-1)^{D-1}$ routers are located at distance D of any origin router. A more relaxed definition is given by the generalized Moore bound, which allows the last *layer* to be only partially filled [22.47].

22.3.4 Dragonfly Topology

So far, topologies have been presented from a very theoretical perspective, mostly ignoring practical aspects

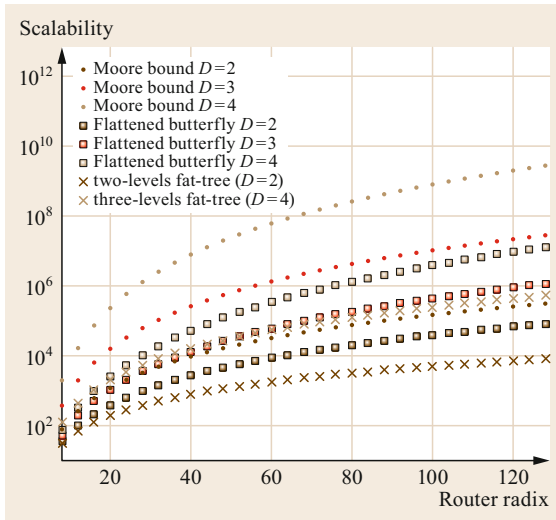


Fig. 22.8 Dots: Maximal theoretical topology scalability according to the Moore bound (22.1); squares: scalability of n -dimensional flattened butterfly; crosses: scalability of two- and three-level fat-trees without oversubscription, for diameter D and router radix r

of field deployments. In particular, the cost difference between electrical and optical links, already introduced in Sect. 22.3.1, *Links*, has not been taken into account. As developed earlier, this cost difference is still relatively important. There is therefore a strong incentive to take the cost of the different links into account when designing a topology, and more specifically, when making oversubscription decisions. This has led interconnect designers to design topologies establishing a clear distinction between long, intercabinet links, and short, intracabinet ones. The dragonfly is the most widespread examples of such constructs.

The dragonfly topology, introduced by Kim et al. [22.48], is a direct topology that strongly relies on the concept of groups. A group is composed of a collection of routers assumed to be physically located in a relatively compact volume (typically within the same cabinet), along with the end-points directly attached to them. Under this assumption, links connecting routers belonging to the same group among themselves, called intragroup links, can remain based on conventional electrical transmission technologies. Links connecting the end-points to the routers are similarly assumed to be electrical. In contrast, links connecting routers across group boundaries, called intergroup links, are optical as they generally traverse long distances.

The dragonfly topology can be seen as hierarchical. Two end-points may be connected to the same router, connected to two distinct routers but part of the same group, or to two distinct routers belonging to different

groups. An optical link is utilized only in the latter case, i.e., only on the upper level of the hierarchy.

When two end-points located in two distinct groups $G1$ and $G2$ communicate, their traffic, after emerging at the *access* router, is first routed within the $G1$ group to a *gateway* router that provides an intergroup link toward $G2$ (first hop). The gateway router sends the traffic over the intergroup link (second hop) and upon arrival in the $G2$ group the traffic is dispatched to the *access* router of the destination end-points (third hop), to finally be distributed to the end-point itself. The dragonfly topology has therefore diameter $D = 3$.

Intragroup Level Dragonfly Connectivity

Within groups, bandwidth is generally made available in large quantities, taking advantage of the relatively low cost of electrical cables. For groups up to 10–15 routers, a full-mesh connectivity can be adopted, whereas for larger groups, a 2-D-flattened butterfly is generally considered.

In the interconnection network of Cray XC series [22.49], taken here as an example to describe a dragonfly group, 96 routers form a group. Each group is internally organized in a 2-D flattened butterfly with lattice $(x = 16) \times (y = 6)$. Each router corresponds to a blade. Sixteen blades form a chassis, which thus represents a *horizontal slice* of the flattened butterfly, and six chassis form a group. Each chassis includes $(16 \times 15)/2 = 120$ electrical links, implemented as wire traces on the chassis backplane, which form the first dimension of the flattened butterfly. A six-chassis group has 720 such links in total. The second dimension of the flattened butterfly is implemented with $16 \times ((5 \times 6)/2) = 240$ links, which are constructed using twin-ax copper cables.

Intergroup Level Dragonfly Connectivity

Groups are connected among themselves by connecting pair of routers belonging to different groups. The intergroup connectivity can take many different aspects, but in general every pair of groups have as at least one direct connection.

Consider a dragonfly composed of g groups with a routers in each group. Further consider that each router has h intergroup links attached to it. To guarantee one link between each pair of groups, a requirement to ensure diameter 3, each group must have $g - 1$ intergroup links, and thus $h \leq (g - 1)/a$. If $h \leq g - 1$, each router can be directly connected to every other group and the dragonfly becomes effectively a 2-D flattened butterfly [22.50].

Setting h to a low value may result in oversubscribed intergroup bandwidth, thus in bottlenecks and overall computational performance penalties for work-

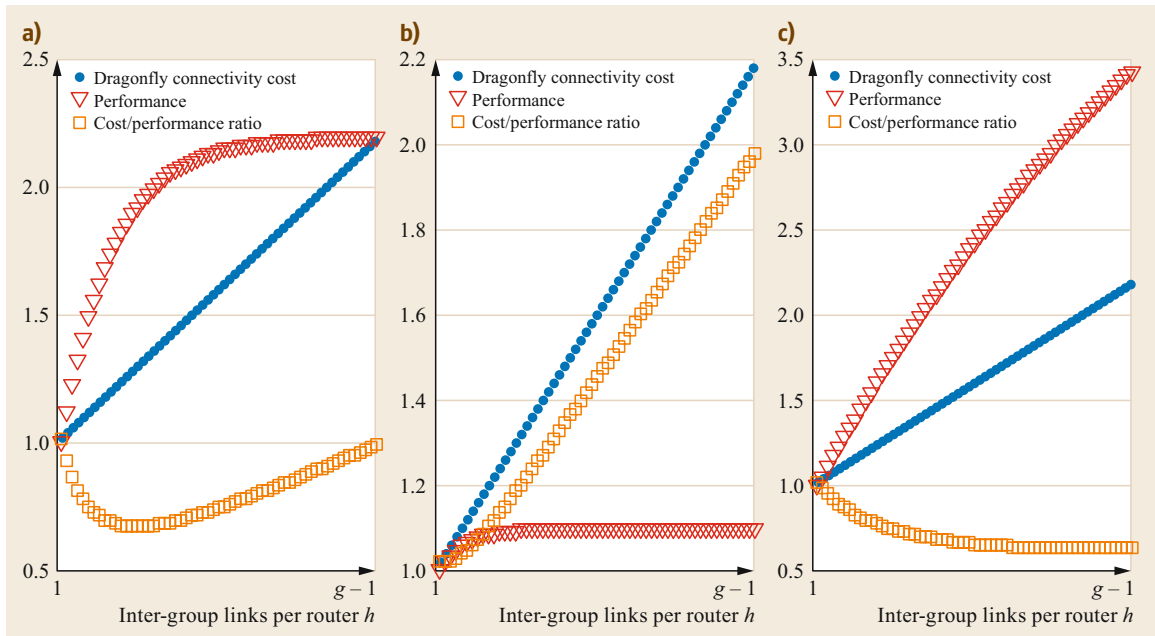


Fig. 22.9a–c Possible scenarios for dragonfly dimensioning. In the three cases, the provisioning of more intergroup links in the dragonfly topology leads to a linear increase of the interconnect cost (blue dots – a reference cost of 1 unit is assumed for a dragonfly with minimal connectivity, i.e., $h = 1$ intergroup link per router), until the point where every router is connected to all other groups through $g - 1$ links. Bandwidth provisioning has, however, different impacts on performance (red triangles). In the case (a), performance grows steady until every router is connected to half of the groups, and saturates hereafter. In the case (b), provisioning more bandwidth only marginally improve performance. In the case (c), performance growth is steady even until reaching full provisioning, suggesting an injection bandwidth bottleneck. Finally, the cost-performance ratio curve (orange squares) indicates which value of h is to advantage: about $h = g/4$ in the case (a), $h = 1$ in the case (b) and $h = g - 1$ in the case (c)

loads involving large amounts of intergroup traffic. In contrast, setting h to a high value has a major impact on global interconnect cost. Dragonflies based HPC architectures are designed so that the selection of h is at the discretion of the customer/operator. Being able to choose h between 1 and $g - 1$ allows the supercomputer vendor to customize the expensive global bandwidth of the dragonfly, and thus, to a large extent, the final cost, to the need and/or budget of the customer. The supercomputer operator, taking into account the traffic locality properties of the workloads that are destined to the system, makes the final call for h . Figure 22.9 shows how h should ideally be selected. The total cost of a system as a function of h , and relative to the $h = 1$ case, is represented by the dots. Cost grows linearly with h . The performance of the system, also relative to the $h = 1$ case, is illustrated by triangles. Performance is expected to grow monotonically with h , as adding bandwidth to a system should not be detrimental to performance, but is also expected to saturate as the computational resources are limited. Finally, the

squares show the cost-performance ratio. Figure 22.9 shows three particular cases. In Fig. 22.9a, performance growth is steady at first but quickly fades as h grows. This results in a sweet spot in the cost-performance ratio curve, which indicates the ideal value of h . In Fig. 22.9b, more intergroup links only limitedly contribute to performance, resulting in an optimal choice of h to be 1. This situation indicates that even with minimal dragonfly connectivity, network resources are generally oversized. Finally, Fig. 22.9c shows the reverse case where cost performance/ratio is continuously improved until $h = g - 1$. This situation is synonym of undersized network resources.

Note that the applicability of this rule is limited by workload execution times prediction capabilities. Also note that the same economical approach can be applied to any part of the interconnect, and more generally, to any design variable of the system. However, changing the amount of memory available inside an end-point, or the bandwidth of the link connecting an end-point to its router, generally requires motherboards to be re-

designed, something impossible to do at procurement time. In contrast, the number of optical links can be provisioned almost independently of the end-point and cabinet design. The dragonfly topology capability to expose an important design variable to the end-user is one of the main factors that drove its adoption.

22.3.5 Higher Networking Layers

Next to offered bandwidth and topologies, HPC interconnects are also characterized by specific operational modes at the link and network layers. The most important of these aspects are mentioned here.

Lossless Flow Control

Intuitively, it is relatively easy to understand that to keep sending packets to a destination that is overwhelmed by previous packets is likely to be counterproductive. In order to prevent gridlocks and subsequent performance collapses [22.51], the amount of traffic that is being injected into a packet-based network must be controlled, something generally described as *flow control*.

Historically, packet-based computer networks have predominantly relied on lossy schemes to achieve flow control. Indeed, the distributed and loosely coupled nature of the Internet has been partly based on the possibility for intermediate routers to drop packets [22.52]: packets that cannot be placed in the queue associated to a link because this queue is full are simply discarded. The recipient, noticing that one packet is missing or that no packets are received at all, informs the sender that congestion is likely underway and that the rate at which data is sent must be reduced.

Lossy flow-control-based networks are well suited for large-scale and poorly structured networks like the Internet but are largely unsuited for HPC. First, the loss of a single packet might cause one of the supercomputer PUs to stall while waiting for the dropped packet to be resent and delivered. The overall parallel performance is very often determined by the performance of the slowest end-point [22.40]. Packet losses can thus have dire consequences in the context of a massively distributed computation. Second, possible packet drops oblige network interfaces to maintain a copy of all packets already sent but not yet acknowledged, which may result in a large memory footprint [22.53]. In addition to these shortcomings directly related to packet loss, lossy flow-control schemes have been shown to lead to poor utilization of bottlenecked links [22.54].

Lossless flow-control schemes have therefore been developed, the most widespread in the HPC context

being the *credit-based* one [22.55]. Each emitter port maintains a counter indicating how many free slots remain available at the corresponding remote reception port. Each time a packet is sent, this counter is decremented. If the counter reaches 0, packet injection is interrupted. Injection resumes upon reception of credits. Whenever a slot is made free at the remote port, this port sends a credit back to the sender port. Upon reception of this credit, the counter is incremented [22.56]. A delay exists between the moment a slot is effectively made free, and the point the sender receives the corresponding credit. The situation where the counter reaches 0 even though free slots are available (therefore with credits on their way) is called a credit stall. To avoid credit stalls, buffer depths at the receiver side must be made proportional to the link latency [22.55]. The use of credit-based flow control is thus limited to local links with relatively low latency.

Routing

A supercomputer being a closed world managed by a single entity, end-points and routers alike can be numbered in independent and consistent ways. This allows us to use much shorter address fields in packets: a 16 bits address field can be used to distinguish up to 65 536 end-points, which is sufficient in most cases (to be compared with 48 bits MAC addresses [22.57], or 128 bits IPv6 ones). This also greatly simplifies routing decisions at routers.

The closed-world vision offered by HPC systems also enables routers, in case of congestion, to adapt their forwarding decisions to exploit the path diversity available in the topology. Figure 22.10 illustrates adaptive routing applied to the 2-D torus and dragonfly cases. Adaptive routing, also known as deflection routing, has been studied in the context of torus [22.58, 59], fat-tree [22.60] and dragonfly-based networks [22.48, 61]. Notable implementations are reported in references [22.28, 30, 62]. Because each router unilaterally and independently decides to deflect part of the traffic to an alternate route, adaptive routing does not lead to minimal congestion levels as if routes were calculated by means of a multicommodity flow formulation [22.63]. In return, routers, when performing adaptive routing, are able to react to congestions in microseconds, if not less. This high reactivity is crucial to prevent counterproductive, day-late-and-dollar-short routing decisions. Recently, adaptive routing has been shown to be competitive compared to per-flow (i.e., per source-destination pair) route optimization and resource management using software defined networking (SDN) [22.63].

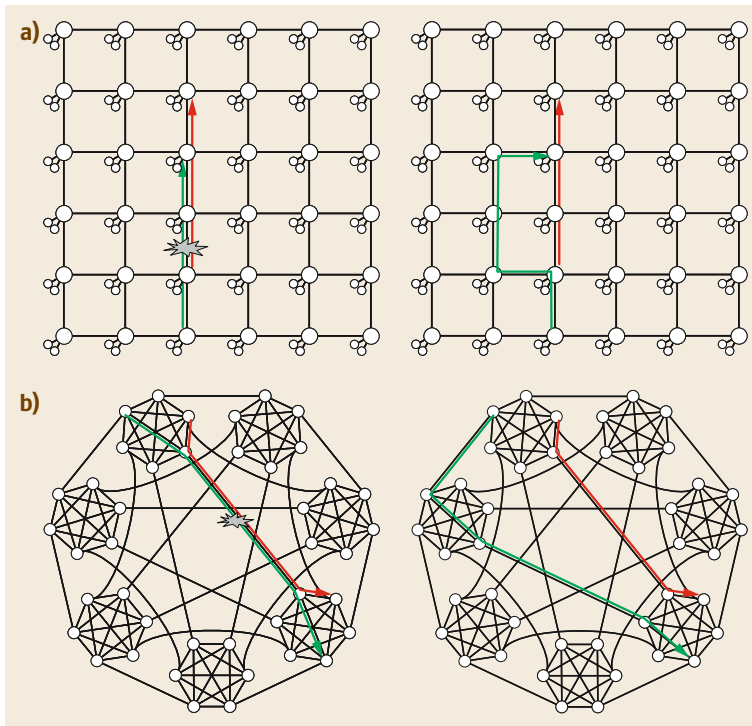


Fig. 22.10a,b Adaptive routing applied to torus (a) and dragonfly (b) topologies. In the first case, the path is lengthened by two hops (from three to five), and in the second case by one hop (from three to four)

22.3.6 Contemporary HPC Networks: Summary

In this section, the characteristics of modern HPC interconnects have been reviewed. The main rules of thumb to retain are the following:

- The standard link-level bandwidth is 100 Gb/s in 2017. This same year some 200 Gb/s products are being introduced, while 400 Gb/s ones are announced for 2018 or 2019.
- Routers routinely provide between 30 and 80 ports, delivering close to 10 Tb/s of switching bandwidth.
- FLOPS installed in a compute node must be multiplied by a factor 0.001–0.01 to obtain the node injection bandwidth (in bytes). A compute node offering 10 TF thus requires between 100 Gb/s and 1 Tb/s of injection bandwidth.
- Due to mismatch between link-level bandwidth (\approx 100 Gb/s) and injection bandwidth, end-points are increasingly being connected through multiple independent links.
- Diameter is considered the important metric for HPC topology, as it guarantees low latency between any pair of end-points. Keeping diameter 2 for 10 000 end-points requires 42 ports at least (Moore bound). A slim-fly [22.34] construct requires 43 ports, a 2-D flattened butterfly 63 and a 2-D fat-tree with no oversubscription, 142 ports.
- Cost difference between electrical and optical cables must be taken into account in the topology design process. Optical links, the most expensive components, must be oversubscribed in priority. The dragonfly topology is specifically thought to ease optical link oversubscription.
- HPC specific flow-control protocol and routing schemes are deployed in many HPC interconnects and permit high utilization and low latency. End-to-end transfer times are often inferior to the microsecond [22.28]. Transmission delay routinely dominates transfer times.

Owing to the very important role that interconnects play in supercomputers, the HPC community has very high expectancies in terms of performance, in particular in term of latency. At the same time, costs and power budgets must be severely controlled.

It is with the rules of this challenging environment in mind that optical technologies for next-generation interconnects must be developed, as addressed in the next section.

22.4 Future of Optics in HPC Interconnects

Most metallic wire-based transmission lines have a cut-off frequency inferior to the GHz. This causes high bandwidth signals with modulated rates superior to the Gb/s to be severely attenuated across distances and suffer signal integrity issues. As supercomputers got larger and required more interconnect bandwidth, designers turned to optics for the longest cables to enable longer reach. The ASCI Purple supercomputer, introduced in 2005, is notably known to be the first system to include optical links [22.64]. Since then, most supercomputers have incorporated optical links. As supercomputers head into the exascale era, and as signaling rates are about to transition from 25 Gb/s to 50 Gb/s to support Tb/s-order link bandwidths required by heavyweight end-points, optical cables are more than ever a constituent of HPC interconnects [22.15]. From its debut in HPC, photonics has also constituted a way to reduce the volume occupied by bulky copper cables, in order to facilitate cooling and simplify maintenance. In 2017, optics is principally seen as a means to limit the bulk around computing equipment, and more precisely to reduce the footprint of metallic traces connecting router ASICs (application-specific integrated circuits) to transceiver connectors (such as QSFP (quad small form-factor pluggable)). Hence, these traces increasingly act as a bottleneck to the total switching bandwidth of routers, which, as discussed in Sect. 22.3.1, *Routers*, need to scale beyond 10 Tb/s. To that aim, solutions to bring tens of Tb/s of *optical bandwidth* directly into the router ASIC or in its direct proximity (e.g., on an underlying interposer die or substrate), and offering large bandwidth densities, are under intense investigation.

In this section, the two main technologies envisioned to realize architectures involving *optically connected router ASICs*, VCSELs (vertical-cavity surface-emitting lasers) with multimode fibers and silicon photonics with single-mode fiber-based links, are reviewed. The capability of these technologies to deliver the required levels of *escape bandwidth density*, in particular, is analyzed. More forward-looking technologies such as plasmonics, optical switching, and free-space optics, are also reviewed and their likelihood to address HPC interconnect challenges analyzed.

22.4.1 VCSEL/MMF Link Technology

The VCSEL technology consists of VCSEL arrays that use both space (i.e., fiber parallel) and wavelength dimensions to increase the total link bandwidth [22.65]. An n -fiber, m -wavelength link requires m dies each con-

sisting of n VCSELs, arranged as a one-dimensional array. Each array is taken from a separate wafer with a specific quantum well epilayer design, dedicated to the emission of a specific wavelength. By adhering to typical coarse wavelength division multiplexing (CWDM) wavelength spacing, the separation between each wavelength of emission of subsequent VCSEL channels is on the order of 25 nm in the range of 850–1100 nm. Note that the VCSEL-specific acronym SWDM (shortwave wavelength division multiplexing) has been introduced for this technology [22.66]. In addition to the m VCSEL array, each link extremity also includes arrays of photodiodes fabricated on the same quantum well epiwafers, with typical aperture of 25 μm , 20–25 GHz 3 dB bandwidth, and 0.8 A/W responsivity. Wavelength multiplexing and demultiplexing is realized by means of passive thin-film filters [22.67]. VCSELs and photodiodes are attached to the carrier also supporting the ASIC, while the filters are attached to the male connector.

An SWDM 200 Gb/s link ($m = 4$ wavelengths and 25 GHz PAM-4 modulation) with 50 m reach has been demonstrated [22.68]. A full design with $n = 6$ fibers per direction, $m = 4$ wavelengths, and performing 25 GHz modulation on each wavelength, resulting in 600 Gb/s full-duplex links, has been reported [22.69].

Packaging, Bandwidth, and Bandwidth Density

VCSEL dies successfully passing wafer-level testing are flip-chip assembled through solder reflow. This assembly process is not damaging and guarantees high yield. The solder reflow technology keeps impedances well controlled as no *hanging* wires are involved. This allows for leaner control circuitries. The injection molded female part of the connector is computer vision aligned to the VCSELs and attached to the carrier with solder reflow as well. This guarantees a satisfying alignment of the female part of the connector with the carrier, and consequently of the VCSELs and photodiodes with the WDM filters and multimode fibers. As of November 2017, fibers were spaced 250 μm apart; 126 μm is considered for the future but is subject to yield issues due to reliable stripping and cleaving of the fiber coating at these narrower dimensions. This limitation is an area of advanced development by fiber manufacturers. Connectors can be clipped in and removed by hand, theoretically allowing manual maintenance operations at the connector level.

The optomechanical package is a major limiting factor for bandwidth density. Connector footprint is

Fig. 22.11 (a) Geometrical analysis of area available for VCSEL around router chip, and of typical wire distances. (b) Impact of substrate size on wire length (*upper x-axis*) and resulting optical bandwidth ▶

typically 1 cm^2 [22.69], which results in a bandwidth density of 6 (Gb/s)/mm^2 in the 600 Gb/s full-duplex example case (12 (Gb/s)/mm^2 if only one direction is considered). Considering a $8 \times 8 \text{ cm}$ large substrate, carrying a $2 \times 2 \text{ cm}$ large ASIC, about 56 cm^2 remain available for VCSEL connectors, enabling $50 \times 0.6 = 30 \text{ Tb/s}$ of bandwidth for the router ASIC. The bandwidth density of 6 (Gb/s)/mm^2 is likely to scale as:

- PAM-4 signaling is used instead of NRZ ($2\times$)
- Signaling rates evolve from 25 Gbd to 50 Gbd ($2\times$)
- Wavelengths are increased from 4 to 6 ($1.5\times$)
- The number of fibers per connector is doubled potentially using $125 \mu\text{m}$ spacing ($2\times$).

Altogether, the technology has the potential to scale to $\approx 7 \text{ Tb/s}$ per per connector, resulting in $\approx 70 \text{ (Gb/s)/mm}^2 = 7 \text{ (Tb/s)/cm}^2$. This would result in extremely compact carriers of roughly $4 \times 4 \text{ cm}$ if 50 Tb/s of ASIC bandwidth are required. Figure 22.11 illustrates how the available optical bandwidth is affected by the substrate size, considering various scenarios. Also visible on the figure is the relationship between substrate size (bottom *x-axis* graduation) and length of metallic wires (top *x-axis* graduation) connecting the ASIC to the VCSELs and photodiodes.

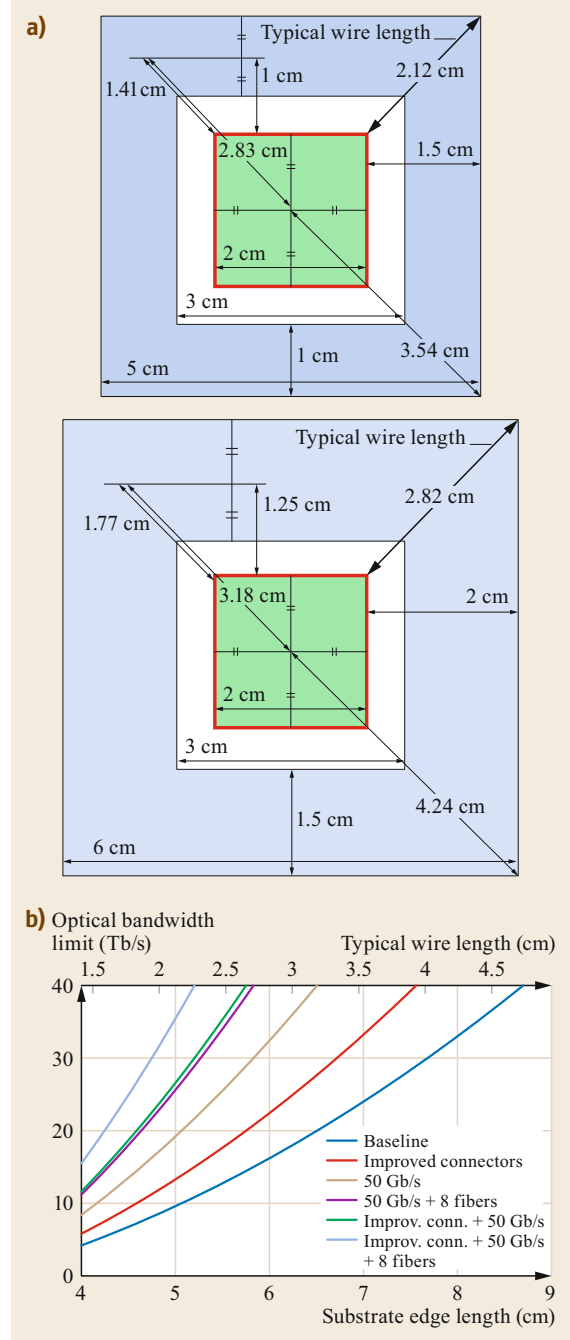
Electrical escape bandwidth densities shall also be considered in conjunction with the optical. Hence, to reach the VCSELs and their associated connectors, electrical signals must

- Escape the main CMOS (complementary metal oxide semiconductor) die to reach the substrate (*vertical escape bandwidth*)
- Escape the chip reticle (*lateral escape bandwidth*).

Considering a $150 \mu\text{m}$ pitch between the solder bumps connecting the ASIC chip to the substrate (45 bumps/mm^2), three bumps per lane (VCSEL drive, photodiode, ground), an ASIC die size of $2 \times 2 \text{ cm}$, and assuming that 20% of the bumps dedicated to IO, the *vertical escape bandwidth* is typically

$$\frac{20\% \times 400 \text{ mm}^2 \times 45 \text{ mm}^{-2} \times 25 \frac{\text{Gb}}{\text{s}}}{3} = 30 \frac{\text{Tb}}{\text{s}}.$$

As for the lateral escape bandwidth, if a $100 \mu\text{m}$ pitch between two wires on the substrate, a 80 mm long chip edge, and four metal layers in substrate are considered,



the total lateral bandwidth is

$$\frac{80 \text{ mm}}{10 \text{ mm}} \times 4 \times \frac{25 \frac{\text{Gb}}{\text{s}}}{3} = 26.6 \frac{\text{Tb}}{\text{s}}.$$

Vertical and lateral escape bandwidths may thus constitute another obstacle to scalability of router bandwidth envelope. They can be scaled if a larger ASIC

chip is considered, although the lateral bandwidth limitation will only scale as the square root of the chip surface. All these calculations are based on a 25 Gb/s bit-rate. Transitioning to 50 Gb/s, for example using PAM-4 modulation, automatically doubles bandwidth.

Transmission Quality and Reach

At 25 Gb/s per wavelength, VCSEL technology can reach distances up to 300 m with standard OM3 fibers. Specialty fibers with optimized modal bandwidth (≈ 5 GHz/km) may be required to carry higher speed signals (50 Gbd) over distances exceeding 50 m, which is sufficient for HPC applications.

With only four to six wavelengths per fiber, wavelength spacing can be kept above 20 nm, which is large enough to tolerate VCSEL wavelength drift due to temperature variations. Thin-film filters can be designed with a band pass wide enough to adequately capture the signal even under changing temperature conditions. Outside of the band of interest, their filtering effect is high and guarantees high crosstalk suppression.

Beyond six wavelengths, the tolerance to temperature variation decreases sharply due to the reduced wavelength spacing. Cascaded losses associated with the multiple reflections taking place within the thin-film filters become significant.

Cost

The VCSEL packaging option allows all components to be aligned and attached through a standard solder reflow process. VCSELs, in particular, are treated as simple *capacitors* by integrated circuit packaging houses. The yield of the assembly process has been shown to be excellent and does not affect costs significantly. VCSELs can be wafer tested and show yields $> 80\%$. Injection molded plastic connectors can be produced at very low cost. For these reasons, it is reasonable to expect the cost of the VCSEL-based solution to drop largely under the critical 1 $\$/(\text{Gb/s})$ mark. This cost is subject to improvement as bandwidth is scaled and/or large quantities are produced.

Power

The power consumption of a directly modulated VCSEL driver is of the order of ≈ 20 mW [22.71], which translates into ≈ 0.8 pJ/bit for 25 Gb/s. Reaching higher modulation speeds requires slightly raised bias currents only, keeping the driving power almost flat. From a pure VCSEL perspective, high symbol rates are thus advantageous for power efficiency.

The VCSEL driving power represents, however, an almost marginal fraction of the total power consumption of an end-to-end link. Pre-emphasis is required before the final driver stage to overcome VCSEL diode

bandwidth and equalization is similarly required after the transimpedance amplification (TIA) and limiting amplifier stages at receiver side. Clock recovery mechanisms must also be accounted for, as well as serialization-deserialization operations. The consumption of each of these elements is of the order of 1 pJ/bit [22.71].

Reliability

The effect of temperatures up to 70 °C as well as of elevated current levels on VCSELs' reliability and wear out have been evaluated. VCSEL lifetimes superior to ten years have been extrapolated. Thermal simulations have shown that temperature should not exceed 70 °C in the close proximity of the VCSELs. Aluminum being a very reactive element, aluminum-free designs should be favored to reach high reliability. VCSELs with active regions comparable to the ones used in pump lasers, showing lifetimes > 25 years, have been designed. The thin-film filters used to demultiplex SWDM signals have routinely been deployed in satellites over the past years and are known to be robust to temperatures well beyond 70 °C, being deposited and annealed at high temperatures. All these elements should concur with excellent reliability figures for VCSEL-based interconnects.

22.4.2 Silicon Photonics Technology

Silicon photonics technology is based on the idea of manipulating light by means of nanoscale structures imprinted in a silicon die using lithographic processes inspired by [22.72, 73] or identical to CMOS [22.74]. After 50 years of silicon semiconductor industrial developments, the electrical properties of silicon are well understood. Silicon photonics builds on this experience: changing electrical properties results in changing optical properties as well. Silicon photonics is currently exploited by several vendors including Mellanox to produce 100 Gb/s active optical cables and short distance transceivers for HPC applications. External modulation is based on Mach-Zehnder interferometers [22.75] or electroabsorption [22.76]. Microring resonators with an integrated PN junction have also been proposed to perform modulation [22.77] and WDM links offering 320 Gb/s on a single fiber have been demonstrated using eight ring resonators and eight ring filters [22.78]. A comprehensive analysis of ring resonator-based links is provided in [22.79].

Packaging, Bandwidth, and Bandwidth Density

Silicon photonics naturally meshes with single-mode optics and thus requires single-mode fibers to be cou-

pled to the optical die. Single-mode optical inputs/outputs allow us to take advantage of the long reach enabled by the low dispersion of single-mode fibers. However, single-mode optics pose a far greater challenge in terms of packaging than multimode optics. An alignment precision of the order of 10% of the mode field diameter, i.e., around one micron, is required, compared to 5–10 μm for multimode fibers in the VCSELs technology.

For packaging solutions that use vertical coupling, grating couplers with approximately 10 micron spot size are the de facto standard, but fiber coupling that is cost effective and scalable is still lacking in industry for these alignment tolerances. Existing solutions still suffer from lack of maturity and result in long, thus expensive, coupling processes, as well as significant optical losses. Advanced, vision-based optical IO connector alignment equipment is under development. Alternative techniques, among others evanescent coupling, have also been investigated [22.70]. Plastic ferrules for connectors are in practice excluded as they cannot tolerate temperature changes observed on the chip edge. Ferrules should thus be made of metal or of a custom material with a coefficient of thermal expansion matching that of silicon. In comparison, edge-coupled solutions that use nitride layers to expand the beam profile and use spot-size converters are also under exploration. This latter technique has the advantage of being polarization insensitive but comes at the expense of larger footprint, increased process steps and requires surface preparation such as dice/cleave or dry etch to expose the correct facet for fiber coupling. Once this facet is prepared, pick-and-place machines can be used to quickly and efficiently align fiber ribbons to the chip to enable high bandwidth density. It should also be noted that such optical chips are thus not a candidate for wafer-level testing and require proxy measurements to make sure the photonic circuits are accurate, which requires the use of vertical grating couplers.

With a projected per-fiber bandwidth of the order of 200–500 Gb/s for silicon photonics, connectors accommodating fiber ribbons with 8–16 fibers are envisioned to reach tens of Tb/s of optical escape bandwidth. Eight connectors each offering eight fibers in each direction (thus 64 fibers in each direction) and 360 Gb/s per fiber (as 24×15 Gb/s) offers for instance 23.04 Tb/s of total bandwidth envelope. The optical escape bandwidth can be scaled to ≈ 50 Tb/s by considering more numerous channels (40 or more, as in DWDM (dense wavelength division multiplexing) applications), higher modulation rates, and/or more advanced modulation formats [22.80]. The number of connectors attached to a chip can also be reasonably scaled. Waveguide pitch is not a limitation as distances as short as 20 μm be-

tween couplers have been demonstrated. As for fiber pitches, assuming 250 μm pitch between two fibers in a cable, 480 fibers can theoretically be attached around the 120 mm long edge of a 3×3 cm interposer. Connector mechanical stability is the eventual limiting factor. Couplers must be separated in connectors of reasonably small size to limit mechanical stress. Consider a 16-fiber connector, whose fibers occupy 4 mm of the edge and with ferrule/attachments that occupy 4 mm as well. Three such 8 mm diameter connectors can be mounted along a > 2.4 cm long edge, four along a > 3.2 cm long one. The maximum number of fibers affixed to a SiP chip can thus be estimated to be 256. This results in 128 full-duplex connections. Combined with a per-fiber bandwidth of 500 Gb/s, a total optical escape bandwidth of 64 Tb/s can be foreseen.

Similar to the VCSEL-based approach, the SiP solution is also subject to electrical escape bandwidth limitations if a dual chip approach is selected and the CMOS drivers are not incorporated into the same die (as shown in Fig. 14 of [22.81], for instance). A ring resonator is typically associated with six control pads: heater control plus ground, diode junction control plus ground, and photodiode plus ground (the photodiode being required to provide feedback for thermal stabilization [22.82]). Since two rings are required per wavelength (neglecting the last wavelength, which requires no ring for filtering), a lane in the SiP solution requires up to 12 bumps. However, since the *driver die* is likely to also be made of silicon, and thus to exhibit similar thermal expansion coefficients, finer bump pitches (40–70 μm) than for the VCSEL solution (150 μm) are allowed. This translates into vertical escape bandwidth in excess of 40 Tb/s:

$$\frac{20\% \times 400 \text{ mm}^2 \text{ (ASIC area devoted to IO)}}{0.07 \times 0.07 \text{ mm}^2 \text{ (area per bump)}} \times \frac{15 \frac{\text{Gb}}{\text{s}} \text{ per lane}}{6 \text{ bumps per lane}} \simeq 40 \frac{\text{Tb}}{\text{s}} .$$

As in the VCSEL case, the area associated with the bumps dominates the silicon real estate requirements of the IO functions. The same holds at the photonic layer: the four pads required to control a ring occupy an area (with 50 μm pitch) of $200 \mu\text{m} \times 200 \mu\text{m}$. This guarantees sufficient spacing between the rings and other optical components. Silicon photonics, being based on single-mode fibers, allows links to span over distances superior to 100 m. Pending appropriate link budgets, quasi error-free operation ($\text{BER} < 10^{-15}$) can be achieved.

Ring-resonator-based modulators should allow energy efficiencies around 1 pJ/bit (not counting additional circuitries). Mach–Zehnder-based modulators are

slightly less efficient. Similar to the VCSEL-based solution, additional circuits are required to operate the transition between the digital and optical formats. However, the silicon photonic solution being able to support a potentially large number of wavelengths, lower bit-rates can be envisioned, which slightly reduces the requirements for pre-emphasis and SERDES (serializer/deserializer). This is expected to result in slightly better energy efficiencies than in the VCSEL case.

22.4.3 Other Forward-Looking Technologies for HPC Interconnects

Optical interconnects for HPC systems based on directly modulated VCSELs combined with MMF (multi-mode fiber) are expected to emerge around the year 2020. These VCSEL-based interconnects will likely be the ones equipping the first exascale systems. Interconnects based on integrated silicon photonics will likely emerge a couple of years later as packaging methods will have matured and cointegrated photonics-specific issues will have been solved.

Beyond this point it is not clear which other photonic technologies can further contribute to HPC interconnect performance. One of the major limitations of photonics resides in the relative bulkiness of the components: a waveguide in silicon has a cross-section close to a tenth of μm^2 . The footprint of a ring resonator in silicon is close to $1000\ \mu\text{m}^2$; that of a photodiode only slightly inferior [22.83] while VCSELs are slightly larger. As wavelengths of interest are in the μm region, guiding structures made of conventional materials cannot be made much smaller than a tenth of μm while devices interacting with light must be at least ten times longer or wider than the wavelength. This is respectively one and three order of magnitude larger than transistors in the most advanced CMOS processes. The important footprint of photonics devices not only limits the bandwidth densities as calculated above. It also prevents scaling of electrical parameters as the resistance (R) and capacitance (C) of electro-optical components in charge of converting electrical signals into optical ones and vice versa [22.84]. The receiver sensitivity, for instance, which is one of the main determinants of the energy efficiency of an optical link as it dictates in most cases the amount of optical power the laser must supply, is strongly determined by the capacitance of the photodiode [22.85].

The footprint drawback of photonics could be alleviated by turning to plasmonics [22.86]. Plasmons represent oscillations of the free electrons present in a metallic volume. Plasmons propagating along the surface of a metallic volume are called surface plasmons. Surface plasmons are very sensitive to environment

changes, can convey energy in a very dense form, and can couple with light. This results in the capacity of plasmonic devices to confine light in subwavelength dimensions [22.87], a feature that can be exploited to minimize footprint [22.88].

Recent research has shown that compact and ultrafast modulators based on plasmonics can be realized [22.89]. The application of plasmonics for photodetectors is reviewed by *Brongersma* [22.84]. Plasmonics have attracted major attention from academia. However, industrial applications of plasmonics, as for instance in HPC interconnects, remain far looking and major gaps preventing adoption must be addressed. In particular, the inclusion of plasmonic materials, such as graphene or polymers, in lithographic processes such as CMOS in a scalable cost-effective way remains to be demonstrated. Moreover, the capability of plasmonic devices to operate at high rates can only be exploited if electrical driving circuits supporting the same rates are available. For speeds above 100 GHz, the use of a lithography process optimized for extremely high frequencies (EHF) such as silicon-germanium is almost mandatory, which drift apart from the high integration goal described earlier in this section. Finally, all demonstrations of communication subsystems involving plasmonics have so far been characterized by high optical losses. Unless gain can be incorporated in plasmonics-based devices [22.90], these losses will severely limit industrial applicability.

A handful of technologies have also been evoked to complement HPC optical interconnects as described in Sect. 22.3. Optical spatial switches have been proposed to reshuffle the fiber connectivity to achieve a better matching between connectivity and traffic matrices over relatively long periods of time [22.91–93]. Technologies underlying the realization of such spatial optical switches are reviewed by *Cheng* et al. [22.94]. Optical switching has also been evoked as a possible replacement for electrical packet switches. However, as summarized in reference [22.15], optical switching at the packet or even flow level is heavily limited by the lack of buffering capabilities in the optical domain.

Instead of reshuffling the topology by means of optical switches, *Fujiwara* et al. [22.95] and *Hu* et al. [22.96] have proposed connecting racks with free-space optical (FSO) links in lieu of optical fibers. By adapting the beam direction, the connectivity can be fully reconfigured. Free-space connections also benefit from a faster propagation time of light in air than in silicon, and from Euclidean distance instead of typical Manhattan routing. Altogether, FSO-based interconnects could reduce the time-of-flight latency by a factor of two. Given that this time-of-flight latency typically represents more than half of the total latency, this can

enable nonnegligible performance gains. FSO links, however, are affected by scintillation and mechanical vibrations. Losses pertaining to this effect must be compensated at the expense of energy efficiency. Moreover, a path to fabrication of FSO links at a price comparable to that of optical links as detailed in Sect. 22.3.1, *Links*, is to be demonstrated. FSO terminals, finally, must be

placed at the top of the rack and can thus be located more than one meter apart from routers or adapters. Communicating data over this distance at 100 Gb/s speeds incurs additional cost and energy consumptions.

The use of FSO links for interchip communications has also been proposed as a way to extend transmission capabilities beyond that of a single fiber [22.97].

22.5 Summary

As high-performance computing applications proliferate, there is a strong global appetite for HPC systems of increasing capability. As shown in Sect. 22.2 of this chapter, the supercomputers required to tackle today's largest computational challenges necessarily rely on chip parallelism to reach tens of petaFLOPS, and soon one or more exaFLOPS, of installed computing power. This places the interconnect at the center of the technical and scientific effort to increase HPC performance.

Building large-scale HPC interconnects is primarily achieved by minimizing the costs, as well as, and increasingly, power consumption. The performance of an interconnect, along with its cost and power consumption, must be related to those of the end-points. These relationships have been detailed in Sect. 22.3. By considering a cost of \$2000 for one installed TF, and a network verbosity factor of 0.01 byte/flop, the cost of the 10 GB/s = 100 Gb/s of interconnection bandwidth required per end-point can be upper bound by several hundreds of dollars. The analyses and explanations provided on the topology level in Sect. 22.3.3 justify this rule of thumb. More importantly, the understanding of simple rules of thumb, and of scalability bounds such as the Moore bound (Sect. 22.3.3, *Moore Bound and Scalability Limits*), allows one to remark that no massive cost savings should be expected from particularly

innovative topologies as long as no oversubscription is applied. Costs can be tapered by means of oversubscription, but tapering ratios should be selected knowing the impact they will have on HPC application performance.

A budget of \$100–200 for 100 Gb/s directly sets a price objective of ≈ 1 \$(/Gb/s), a figure that is thus far out of reach for optical technologies. The most immediate and stringent requirement of HPC in terms of optical technologies thus consists today of ways and means to rationalize transceiver development costs. As detailed in Sect. 22.4, multiwavelength VCSEL-based CWDM links and silicon photonic integration are among the main technologies envisioned to attain the cost and power efficiency metrics dictated by HPC trends.

Highly integrated optical technologies are also required to allow total switching bandwidths of packet routers to scale beyond 10 Tb/s. High FLOPS density compute nodes totaling tens of TFs have an urgent need for link bandwidths of 400 Gb/s and beyond. At such rates, the reach of passive cables will likely be reduced to ≈ 1 m, obliging packet routers to communicate optically on most of their ports. Here as well, VCSEL/MMF and silicon photonics technologies are the best positioned to solve the packet router bandwidth density challenge.

References

- 22.1 J.S. Dietrich: Cosmic cubism, *Eng. Sci.* **47**(4), 17–20 (1984)
- 22.2 J. Gao, H. Cheng, H.C. Wu, G. Liu, E. Lau, L. Yuan, C. Krause: Thunderbolt Interconnect - Optical and copper, *J. Lightwave Technol.* **35**(15), 3125–3129 (2017)
- 22.3 T. Mikolov, K. Chen, G. Corrado, J. Dean: Efficient estimation of word representations in vector space, *arXiv:1301.3781 [cs.CL]* (2013)
- 22.4 D.A. Reed, J. Dongarra: Exascale computing and big data, *Commun. ACM* **58**(7), 56–68 (2015)
- 22.5 J.S. Vetter: Contemporary high performance computing. In: *Contemporary High Performance Computing: From Petascale Toward Exascale* (Chapman & Hall, CRC, New York 2013)
- 22.6 N.R. Adiga, G. Almasi, G.S. Almasi, Y. Aridor, R. Barik, D. Beece, R. Bellofatto, G. Bhanot, R. Bickford, M. Blumrich, A.A. Bright, J. Brunheroto, C. Cascaval, J. Castanos, W. Chan, L. Ceze, P. Co-teus, S. Chatterjee, D. Chen, G. Chiu, T.M. Cipolla, P. Crumley, K.M. Desai, A. Deutsch, T. Domany, M.B. Dombrowa, W. Donath, M. Eleftheriou, C. Erway, J. Esch, B. Fitch, J. Gagliano, A. Gara, R. Garg, R. Germain, M.E. Giampapa, B. Gopalsamy, J. Gun-nels, M. Gupta, F. Gustavson, S. Hall, R.A. Haring, D. Heidel, P. Heidelberger, L.M. Herger, D. Hoenicke,

- R.D. Jackson, T. Jamal-Eddine, G.V. Kopcsay, E. Krevat, M.P. Kurhekar, A.P. Lanzetta, D. Lieber, L.K. Liu, M. Lu, M. Mendell, A. Misra, Y. Moatti, L. Mok, J.E. Moreira, B.J. Nathanson, M. Newton, M. Ohmacht, A. Oliner, V. Pandit, R.B. Pudota, R. Rand, R. Regan, B. Rubin, A. Ruehli, S. Rus, R.K. Sahoo, A. Sanomiya, E. Schenfeld, M. Sharma, E. Shmueli, S. Singh, P. Song, V. Srinivasan, B.D. Steinmacher-Burow, K. Strauss, C. Surovic, R. Swetz, T. Takken, R.B. Tremaine, M. Tsao, A.R. Umamaheshwaran, P. Verma, P. Vranas, T.J.C. Ward, M. Wazlowski, W. Barrett, C. Engel, B. Drechsel, B. Hilgart, D. Hill, F. Kasemkhani, D. Krolak, C.T. Li, T. Liebsch, J. Marcella, A. Muff, A. Okomo, M. Rouse, A. Schram, M. Tubbs, G. Ulsh, C. Wait, J. Wittrup, M. Bae, K. Dockser, L. Kissel, M.K. Seager, J.S. Vetter, K. Yates: An Overview of the BlueGene/L supercomputer. In: *Proc. ACM Conf. Supercomput.* (2002)
- 22.7 J. Dongarra: Report on the Sunway TaihuLight System, Univ. Tennessee Comput. Sci. Rep. UT-EECS-16-742 (2016)
- 22.8 R. Dolbeau: Theoretical Peak FLOPS per instruction set on less conventional hardware, <http://www.dolbeau.name/dolbeau/publications/peak-alt.pdf> (2016)
- 22.9 TOP500.org: Top 500 supercomputer sites, <https://www.top500.org/> (2019)
- 22.10 The High Performance Conjugate Gradients (HPCG) Benchmark Project: <http://www.hpcg-benchmark.org/>
- 22.11 S. Rumley, D. Nikolova, R. Hendry, Q. Li, D. Calhoun, K. Bergman: Silicon photonics for exascale systems, *J. Lightwave Technol.* **33**(3), 547–562 (2015)
- 22.12 Graph 500: Large-scale benchmarks, <https://graph500.org/> (2017)
- 22.13 R.C. Murphy, K.B. Wheeler, B.W. Barrett, J.A. Ang: Introducing the graph 500. In: *Proc. CUG* (2010)
- 22.14 E. Joseph, S. Conway: *Major Trends in the World-wide HPC Market* (IDC, Framingham 2017)
- 22.15 S. Rumley, M. Bahadori, R. Polster, S.D. Hammond, D.M. Calhoun, K. Wen, A. Rodrigues, K. Bergman: Optical interconnects for extreme scale computing systems, *Parallel Comput.* **64**(Suppl. C), 65–80 (2017)
- 22.16 U.S. Energy Information Administration: Electricity monthly update, https://www.eia.gov/electricity/monthly/update/wholesale/protect_markets.php (2019)
- 22.17 J.E. Thornton: The CDC 6600 project, *Ann. Hist. Comput.* **2**(4), 338–348 (1980)
- 22.18 A. Aiken, U. Banerjee, A. Kejariwal, A. Nicolau: Overview of ILP architectures. In: *Instruction Level Parallelism* (Springer, New York 2016) pp. 9–42
- 22.19 J.S. Vetter, B.R. de Supinski, L. Kissel, J. May, S. Vaidya: Evaluating high-performance computers, *Concurr. Comput.* **17**(10), 1239–1270 (2005)
- 22.20 R. Espasa, M. Valero, J.E. Smith: Vector architectures: Past, present and future. In: *Proc. Int. Conf. Supercomp.* (1998) pp. 425–432
- 22.21 A.H. Karp, M. Heath, D. Heller, H. Simon: 1994 Gordon Bell Prize winners, *Computer* **28**(1), 68–74 (1995)
- 22.22 R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, A.R. LeBlanc: Design of ion-implanted MOS-FET's with very small physical dimensions, *IEEE J. Solid-State Circuits* **9**(5), 256–268 (1974)
- 22.23 M.M. Waldrop: The chips are down for Moores law, *Nature* **530**, 144–147 (2016)
- 22.24 N. Loubet, T. Hook, P. Montanini, C.W. Yeung, S. Kanakasabapathy, M. Guillom, T. Yamashita, J. Zhang, X. Miao, J. Wang, A. Young, R. Chao, M. Kang, Z. Liu, S. Fan, B. Hamieh, S. Sieg, Y. Mignot, W. Xu, S.C. Seo, J. Yoo, S. Mochizuki, M. Sankarapandian, O. Kwon, A. Carr, A. Greene, Y. Park, J. Frougier, R. Galatage, R. Bao, J. Shearer, R. Conti, H. Song, D. Lee, D. Kong, Y. Xu, A. Arceo, Z. Bi, P. Xu, R. Muthinti, J. Li, R. Wong, D. Brown, P. Oldiges, R. Robison, J. Arnold, N. Felix, S. Skordas, J. Gaudiello, T. Standaert, H. Jagannathan, D. Corliss, M.H. Na, A. Knorr, T. Wu, D. Gupta, S. Lian, R. Divakaruni, T. Gow, C. Labelle, S. Lee, V. Paruchuri, H. Bu, M. Khare: Stacked nanosheet gate-all-around transistor to enable scaling beyond FinFET. In: *Symp. VLSI Technol.* (2017) pp. T230–T231
- 22.25 IEEE: IRDS More Moore white paper, https://irds.ieee.org/images/files/pdf/2016/protect_MM.pdf (2016)
- 22.26 D. Foley, J. Danskin: Ultra-performance Pascal GPU and NVlink interconnect, *IEEE Micro* **37**(2), 7–17 (2017)
- 22.27 S. Derradji, T. Palfer-Sollier, J.P. Panziera, A. Poudes, F.W. Ato: The BXI interconnect architecture. In: *IEEE 23rd Proc. Ann. Symp. High Perform. Interconnects* (2015) pp. 18–25
- 22.28 D. Chen, N.A. Easley, P. Heidelberger, R.M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D.L. Satterfield, B. Steinmacher-Burow, J.J. Parker: The IBM Blue Gene/Q interconnection network and message unit. In: *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.* (2011), Art. 26
- 22.29 S. Scott, D. Abts, J. Kim, W.J. Dally: The BlackWidow high-radix Clos network, *ACM SIGARCH Comput. Archit. News* **34**(2), 16–28 (2006)
- 22.30 G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, J. Reinhard: Cray Cascade: A scalable HPC system based on a Dragonfly network. In: *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.* (2012), Art. 103
- 22.31 Cray: Cray® XC50™ supercomputer, <https://www.cray.com/sites/default/files/Cray-XC50-NVIDIA-Tesla-P100-GPU-Accelerator-Blade.pdf> (2017)
- 22.32 Mellanox: InfiniBand switch systems, http://www.mellanox.com/page/switch/protect_systems/protect_overview (2019)
- 22.33 S.H. Russ: Differential signaling. In: *Signal Integrity: Applied Electromagnetics and Professional Practice* (Springer, Berlin 2016) pp. 101–109
- 22.34 M. Besta, T. Hoefler: Slim Fly: A cost effective low-diameter network topology. In: *Proc. Int. Conf.*

- High Perform. Comput. Netw. Storage Anal.* (2014) pp. 348–359
- 22.35 C.A. Thraskias, E.N. Lallas, N. Neumann, L. Schares, B.J. Offrein, R. Henker, D. Plettemeier, F. Ellinger, J. Leuthold, I. Tomkos: Survey of photonic and plasmonic interconnect technologies for intra-datacenter and high-performance computing communications, *IEEE Commun. Surv. Tutor.* **20**(4), 2758–2783 (2018)
- 22.36 S. Rumley, R.P. Polster, S.D. Hammond, A.F. Rodrigues, K. Bergman: End-to-end modeling and optimization of power consumption in HPC interconnects. In: *45th Int. Conf. Parallel Process. Workshops* (2016) pp. 133–140, <https://doi.org/10.1109/ICPPW.2016.33>
- 22.37 V. Marjanović, J. Gracia, C.W. Glass: Performance modeling of the HPCG benchmark. In: *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation. PMBS 2014*, Lecture Notes in Computer Science, Vol. 8966, ed. by S. Jarvis, S. Wright, S. Hammond (Springer, Cham 2015) pp. 172–192
- 22.38 G. Michelogiannakis, K.Z. Ibrahim, J. Shalf, J.J. Wilke, S. Knight, J.P. Kenny: APHiD: Hierarchical task placement to enable a tapered fat tree topology for lower power and cost in HPC networks. In: *7th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.* (2017) pp. 228–237
- 22.39 S. Lammel, F. Zahn, H. Fröning: SONAR: Automated communication characterization for HPC applications. In: *High Performance Computing. ISC High Performance 2016*, Lecture Notes in Computer Science, Vol. 9945, ed. by M. Tauber, B. Mohr, J. Kunkel (Springer, Cham 2016) pp. 98–114
- 22.40 J. Dean, L.A. Barroso: The tail at scale, *Commun. ACM* **56**(2), 74–80 (2013)
- 22.41 N. Jain, A. Bhatele, S. White, T. Gamblin, L.V. Kale: Evaluating HPC networks via simulation of parallel workloads. In: *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.* (2016) pp. 154–165
- 22.42 N. Jain, A. Bhatele, L.H. Howell, D. Böhme, I. Karlin, E.A. León, M. Mubarak, N. Wolfe, T. Gamblin, M.L. Leininger: Predicting the performance impact of different fat-tree configurations. In: *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.* (2017), Art. 50
- 22.43 G. Kathareios, C. Minkenbergh, B. Prisacari, G. Rodriguez, T. Hoefler: Cost-effective diameter-two topologies: Analysis and evaluation. In: *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.* (2017), Art. 36
- 22.44 Y. Ajima, Y. Takagi, T. Inoue, S. Hiramoto, T. Shimizu: The tofu interconnect. In: *IEEE 19th Ann. Symp. High Perform. Interconnects* (2011) pp. 87–94
- 22.45 J. Kim, W.J. Dally, D. Abts: Flattened butterfly: A cost-efficient topology for high-radix networks, *ACM SIGARCH Comput. Archit. News* **35**(2), 126–137 (2007)
- 22.46 J.H. Ahn, N. Binkert, A. Davis, M. McLaren, R.S. Schreiber: HyperX: Topology, routing, and packaging of efficient large-scale networks. In: *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.* (2009), Art. 41
- 22.47 S. Rumley, M. Glick, S.D. Hammond, A. Rodrigues, K. Bergman: Design methodology for optimizing optical interconnection networks in high performance systems. In: *High Performance Computing. ISC High Performance 2015*, Lecture Notes in Computer Science, Vol. 9137, ed. by J. Kunkel, T. Ludwig (Springer, Cham 2015) pp. 454–471
- 22.48 J. Kim, W.J. Dally, S. Scott, D. Abts: Technology-driven, highly-scalable dragonfly topology, *ACM SIGARCH Comput. Archit. News* **36**(3), 77–88 (2008)
- 22.49 B. Alverson, E. Froese, L. Kaplan, D. Roweth: Cray XC series network, WP-Aries01-1112 (2012)
- 22.50 M.Y. Teh, J.J. Wilke, K. Bergman, S. Rumley: Design space exploration of the dragonfly topology. In: *High Performance Computing. ISC High Performance 2017*, Lecture Notes in Computer Science, Vol. 10524, ed. by J. Kunkel, R. Yokota, M. Tauber, J. Shalf (Springer, Cham 2017) pp. 57–74
- 22.51 P. Gevros, J. Crowcroft, P. Kirstein, S. Bhatti: Congestion control mechanisms and the best effort service model, *IEEE Netw.* **15**(3), 16–26 (2001)
- 22.52 V. Jacobson: Congestion avoidance and control, *SIGCOMM Comput. Commun. Rev.* **18**(4), 314–329 (1988)
- 22.53 P. Grun: *Introduction to InfiniBand for End Users* (InfiniBand Trade, Beaverton 2010)
- 22.54 A. Falk, T. Faber, J. Bannister, A. Chien, R. Grossman, J. Leigh: Transport protocols for high performance, *Commun. ACM* **46**(11), 42–49 (2003)
- 22.55 W. Dally, B. Towles: *Principles and Practices of Interconnection Networks* (Morgan Kaufmann, New York 2003)
- 22.56 T.M. Pinkston, J. Duato: Appendix F: Interconnection networks. In: *Computer Architecture: A Quantitative Approach*, 5th edn., ed. by J.L. Hennessy, D.A. Patterson (Morgan Kaufmann, Waltham 2011)
- 22.57 IEEE Standards Association: Standard group MAC addresses: A tutorial guide, <https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tutorials/macgrp.pdf> (2019)
- 22.58 S.L. Scott, G.M. Thorson: The cray t3e network: Adaptive routing in a high performance 3d torus. In: *Proc. Symp. HOT Interconnects IV* (1996) pp. 147–156
- 22.59 A. Singh, W.J. Dally, A.K. Gupta, B. Towles: GOAL: A load-balanced adaptive routing algorithm for torus networks. In: *Proc. 30th Ann. Int. Symp. Comput. Archit.* (2003) pp. 194–205
- 22.60 J. Kim, W.J. Dally, D. Abts: Adaptive routing in high-radix cros network. In: *Proc. ACM/IEEE Conf. Supercomput.* (2006), Art. 92
- 22.61 N. Jiang, J. Kim, W.J. Dally: Indirect adaptive routing on large scale interconnection networks, *ACM SIGARCH Comput. Archit. News* **37**(3), 220–231 (2009)
- 22.62 R. Alverson, D. Roweth, L. Kaplan: The Gemini system interconnect. In: *IEEE 18th Symp. High Perform. Interconnects* (2010) pp. 83–87
- 22.63 P. Faizian, M.A. Mollah, Z. Tong, X. Yuan, M. Lang: A comparative study of SDN and adaptive routing on

- dragonfly networks. In: *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.* (2017), Art. 51
- 22.64 B. Barney: Using ASC Purple, <https://computing.llnl.gov/tutorials/purple/index.html> (2019)
- 22.65 S. Benjamin, K. Hasharoni, A. Maman, S. Stepanov, M. Mesh, H. Luesebrink, R. Steffek, W. Pleyer, C. Stömmmer: 336-Channel electro-optical interconnect: Underfill process improvement, fiber bundle and reliability results. In: *IEEE 64th Electron. Compon. Technol. Conf.* (2014) pp. 1021–1027
- 22.66 SWDM Alliance: <http://www.swdm.org/>
- 22.67 T. Saeki, S. Sato, M. Kurokawa, A. Moto, M. Suzuki, K. Tanaka, K. Tanaka, N. Ikoma, Y. Fujimura: 100 Gbit/s compact transmitter module integrated with optical multiplexer. In: *IEEE Photonics Conf.* (2013) pp. 307–308
- 22.68 T.N. Huynh, F. Doany, D.M. Kuchta, D. Gazula, E. Shaw, J. O'Daniel, J. Tatum: 4×50Gb/s NRZ short-wave-wavelength division multiplexing VCSEL link over 50m multimode fiber. In: *Opt. Fiber Commun. Conf. Exhib.* (2017), <https://doi.org/10.1364/OFC.2017.Tu2B.5>
- 22.69 M.R.T. Tan, P. Rosenberg, W.V. Sorin, B. Wang, S. Mathai, G. Panotopoulos, G. Rankin: Universal photonic interconnect for data centers, *J. Lightwave Technol.* **36**(2), 175–180 (2018)
- 22.70 L. Carroll, J.-S. Lee, C. Scarcella, K. Gradkowski, M. Duperron, H. Lu, Y. Zhao, C. Eason, P. Morrissey, M. Rensing, S. Collins, H.Y. Hwang, P. O'Brien: Photonic packaging: Transforming silicon photonic integrated circuits into photonic devices, *Appl. Sci.* **6**(12), 426 (2016)
- 22.71 N. Kohmu, T. Takai, N. Chujo, H. Arimoto: A 25.78-Gbit/s × 4-ch active optical cable with ultra-compact form factor for high-density optical interconnects, *Appl. Sci.* **8**(1), 137 (2018)
- 22.72 Y. Arakawa, T. Nakamura, Y. Urino, T. Fujita: Silicon photonics for next generation system integration platform, *IEEE Commun. Mag.* **51**(3), 72–77 (2013)
- 22.73 P. Dumon: Towards foundry approach for silicon photonics: Silicon photonics platform ePIXfab, *Electron. Lett.* **45**(1), 581–582 (2009)
- 22.74 M.T. Wade, J.M. Shainline, J.S. Orcutt, C. Sun, R. Kumar, B. Moss, M. Georgas, R.J. Ram, V. Stojanović, M.A. Popović: Energy-efficient active photonics in a zero-change, state-of-the-art CMOS process. In: *Opt. Fiber Commun. Conf.* (2014), <https://doi.org/10.1364/OFC.2014.Tu2E.7>
- 22.75 L. Liao, A. Liu, D. Rubin, J. Basak, Y. Chetrit, H. Nguyen, R. Cohen, N. Izhaky, M. Paniccia: 40 Gbit/s silicon optical modulator for high-speed applications, *Electron. Lett.* **43**(22), 1196 (2007)
- 22.76 Y. Tang, H.-W. Chen, S. Jain, J.D. Peters, U. Westergren, J.E. Bowers: 50 Gb/s hybrid silicon traveling-wave electroabsorption modulator, *Opt. Express* **19**(7), 5811–5816 (2011)
- 22.77 Q. Xu, B. Schmidt, S. Pradhan, M. Lipson: Micrometre-scale silicon electro-optic modulator, *Nature* **435**(7040), 325–327 (2005)
- 22.78 Y. Liu, R. Ding, Q. Li, Z. Xuan, Y. Li, Y. Yang, A.E. Lim, P.G. Lo, K. Bergman, T. Baehr-Jones, M. Hochberg: Ultra-compact 320 Gb/s and 160 Gb/s WDM transmitters based on silicon microrings. In: *Opt. Fiber Commun. Conf.* (2014), <https://doi.org/10.1364/OFC.2014.Th4G.6>
- 22.79 M. Bahadori, S. Rumley, D. Nikolova, K. Bergman: Comprehensive design space exploration of silicon photonic interconnects, *J. Lightwave Technol.* **34**(12), 2975–2987 (2016)
- 22.80 M. Bahadori, S. Rumley, R. Polster, A. Gazman, M. Traverso, M. Webster, K. Patel, K. Bergman: Energy-performance optimized design of silicon photonic interconnection networks for high-performance computing. In: *Design Autom. Test Eur. Conf. Exhib.* (2017) pp. 326–331
- 22.81 A.V. Krishnamoorthy, O. Torudbakken, S. Müller, A. Srinivasan, P. Decker, H. Opheim, J.E. Cunningham, X. Zheng, M. Dignum, K. Raj: From chip to cloud: Optical interconnects in engineered systems for the enterprise. In: *IEEE Opt. Interconnects Conf.* (2016) pp. 34–35
- 22.82 K. Padmaraju, D.F. Logan, T. Shiraishi, J.J. Ackert, A.P. Knights, K. Bergman: Wavelength locking and thermally stabilizing microring resonators using dithering signals, *J. Lightwave Technol.* **32**(3), 505–512 (2014)
- 22.83 M. Piels, J.E. Bowers: 40 GHz Si/Ge uni-traveling carrier waveguide photodiode, *J. Lightwave Technol.* **32**(20), 3502–3508 (2014)
- 22.84 M.L. Brongersma: Plasmonic photodetectors, photovoltaics, and hot-electron devices, *Proc. IEEE Inst. Electr. Electron Eng.* **104**(12), 2349–2361 (2016)
- 22.85 D.A.B. Miller: Attojoule optoelectronics for low-energy information processing and communications, *J. Lightwave Technol.* **35**(3), 346–396 (2017)
- 22.86 S.A. Maier: *Plasmonics: Fundamentals and Applications* (Springer, New York 2007)
- 22.87 D.K. Mynbaev, V. Sukharenko: Plasmonics for optical communications: The use of graphene for optimizing coupling efficiency. In: *Int. Caribb. Conf. Devices Circuits Syst.* (2014), <https://doi.org/10.1109/ICDCS.2014.7016180>
- 22.88 M.L. Brongersma, V.M. Shalaev: The case for plasmonics, *Science* **328**(5977), 440–441 (2010)
- 22.89 M. Ayata, Y. Fedoryshyn, W. Heni, B. Baeuerle, A. Josten, M. Zahner, U. Koch, Y. Salamin, C. Hoessbacher, C. Haffner, D.L. Elder, L.R. Dalton, J. Leuthold: High-speed plasmonic modulator in a single metal layer, *Science* **358**(6363), 630–632 (2017)
- 22.90 D. Dai, Y. Shi, S. He, L. Wosinski, L. Thylen: Gain enhancement in a hybrid plasmonic nano-waveguide with a low-index or high-index gain medium, *Opt. Express* **19**(14), 12925–12936 (2011)
- 22.91 S. Kamil, L. Olike, A. Pinar, J. Shalf: Communication requirements and interconnect optimization for high-end scientific applications, *IEEE Trans. Parallel Distrib. Syst.* **21**(2), 188–202 (2010)
- 22.92 K. Wen, P. Samadi, S. Rumley, C.P. Chen, Y. Shen, M. Bahadroi, K. Bergman, J. Wilke: Flexfly: Enabling a reconfigurable dragonfly through silicon photonics. In: *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.* (2016) pp. 166–177

- 22.93 C. Minkenberg, G. Rodriguez, B. Prisacari, L. Schares, P. Heidelberger, D. Chen, C. Stunkel: Performance benefits of optical circuit switches for large-scale dragonfly networks. In: *Opt. Fiber Commun. Conf.* (2016), <https://doi.org/10.1364/OFC.2016.W3J.3>
- 22.94 Q. Cheng, S. Rumley, M. Bahadori, K. Bergman: Optical technologies for data centers: A review, *Optica* **5**(11), 1354 (2018)
- 22.95 I. Fujiwara, M. Koibuchi, T. Ozaki, H. Matsutani, H. Casanova: Augmenting low-latency HPC network with free-space optical links. In: *IEEE 21st Int. Symp. High Perform. Comput. Archit.* (2015) pp. 390–401
- 22.96 Y. Hu, I. Fujiwara, M. Koibuchi: Job mapping and scheduling on free-space optical networks, *IEICE Trans. Inform. Syst.* **E99.D**(11), 2694–2704 (2016)
- 22.97 J.M. Kahn, D.A.B. Miller: Communications expands its space, *Nat. Photonics* **11**(1), 5–8 (2017)

Sébastien Rumley

Dept. of Electrical Engineering
Columbia University
New York, NY, USA
sebastien.rumley@olympic.ch



Sébastien Rumley received his MS and PhD degrees in Communication Systems from EPFL, Switzerland, in 2005 and 2011. He is currently Research Scientist with the Lightwave Research Laboratory, Columbia University, New York. His research interests include cross-scale modeling and optimization of large-scale interconnection networks, with particular focus on optical technologies and silicon photonics.

Keren Bergman

Dept. of Electrical Engineering
Columbia University
New York, NY, USA
bergman@ee.columbia.edu



Keren Bergman received a BS from Bucknell University in 1988 an MS in 1991, and a PhD in 1994 from MIT, all in Electrical Engineering. Bergman leads the Lightwave Research Laboratory encompassing multiple cross-disciplinary programs at the intersection of computing and photonics. Bergman serves on the Leadership Council (AIM) Photonics. She is a Fellow of the Optical Society of America (OSA) and IEEE.

M. Ashkan Seyedi



Hewlett Packard Labs
HPE
Palo Alto, CA, USA
ashkan.seyedi@hpe.com

Ashkan Seyedi received a dual Bachelor's in Electrical and Computer Engineering from the University of Missouri-Columbia and a PhD from the University of Southern California working on photonic crystal devices, high-speed nanowire photodetectors, efficient white LEDs, and solar cells. While at Hewlett Packard Enterprise as a research scientist, he has been working on developing high-bandwidth, efficient optical interconnects for exascale and high-performance computing applications.

Marco Fiorentino



Hewlett Packard Labs
HPE
Palo Alto, CA, USA
marco.fiorentino@hpe.com

Marco Fiorentino is a Distinguished Technologist at Hewlett Packard Labs. Before joining HPE in 2005 he worked at Northwestern University, the University of Rome, and MIT. Marco received a PhD in Physics from the University of Napoli, Italy in 2000, working on quantum optics. He is currently working on silicon photonics technologies for computercom. He is a Senior Member of the Optical Society of America.