Software and Hardware Techniques for Power-Efficient HPC Networking

Although most large-scale systems are designed with the network as a central component, the interconnection network's energy consumption has received little attention. However, several software and hardware approaches can increase the interconnection network's power efficiency by using the network more efficiently or using throttling bandwidths to reduce the power consumption of unneeded resources.

Power-aware or "green" HPC is receiving growing attention—not least due to the eye-opening bottom line on the energy bill for most HPC data centers. Other factors include steadily growing performance requirements in many application areas, the stagnating performance-per-watt ratio of general-purpose servers,[1] and—as Figure 1 shows—the inexorably growing costs of electrical energy.

A simple calculation, based on real data from an SC08 Cluster Challenge Competition entry shows that energy costs over a server's four-year life period already make up a significant fraction of its total cost of ownership. Typical servers cost approximately $3,000, including basic networking costs for a small-scale system. Such a system (including networking equipment) draws between 150 and 350 watts depending on its load. So, under a full load, assuming 11 cents per kilowatt hour (kWh), the total power consumption adds up to approximately $1,350 over four years, which is 45 percent of the cost of the server. For a commodity cluster system with 512 compute nodes, the cost would be $14,400 per month.

1521-9615/10/$26.00 © 2010 IEEE
Copublished by the IEEE CS and the AIP

Torsten Hoefler
National Center for Supercomputing Applications

The communication network, or *interconnect*, forms the backbone of every large-scale computing system. Most multipurpose large-scale computing systems are actually built around the communication network and its topology. The network topology describes how network elements, such as switches or endpoints (nodes), connect to each other. Thus, we distinguish two important parameters of each interconnection network: the network technology (such as Myrinet, InfiniBand, or Ethernet) and the network topology (such as fat-tree, torus, or Kautz graph). As we describe here, both play an important role in the communication network's growing importance in green HPC.

Problem Overview

In our model system—a small-scale setting with a single switch and 14 clients—the network equipment consumed less than 10 percent of the total idle power, equaling $1,440 per month. However, network power consumption is expected to grow at scale, and several sources report that their large-scale systems consume between 15 and 20 percent of the total idle power. Another interesting effect is that our system's power consumption increased by more than 20 percent when running a simple communication benchmark. We also observed that, regardless of the load, we can achieve a power savings of up to 5 percent by choosing the right interconnect.

The development of power-saving techniques for other elements in servers, such as the CPU (turning off or throttling cores), memory (new technologies, such as "not and," or NAND, Flash[3]), or more efficient power supplies and blade technologies, is progressing quickly. However, similar techniques in networking haven't yet reached wide adoption. Thus, the relative importance of power-savings in the network is likely to grow in the near future.

In addition, steadily growing computing demands to perform large-scale simulations at the forefront of science will soon require extreme-scale computers. As we move from peak-petaflop machines to sustained-petaflop to peak-exaflop computers, we're likely to hit the "power wall," which will require a completely new way of thinking.

There are several approaches for proceeding to exascale under obvious energy limitations. The most successful approach to energy savings is to design special hardware.[4,5] However, this often means high design costs and relatively limited use.

A more versatile, but still limited approach is to use heterogeneous or asymmetric multiprocessing and accelerators. This typically requires fundamental changes in algorithm design and implementation to exploit the specialized architectures' power. Such specialized solutions are gaining importance, but the significant investments necessary to build large-scale computing systems require that those resources are versatile and can run numerous different applications. Also, some application classes can use holistic approaches to the energy problem, such as the Green Destiny project[6] or Blue Gene.[7] However, the network is often a weak part of such systems.

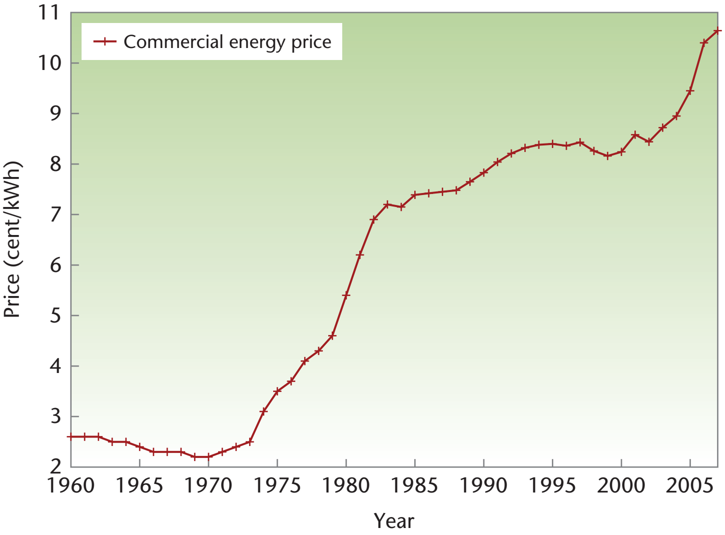

Figure 1. Development of US electrical energy prices according to the US Energy Information Administration's Monthly Energy Review for October 2008.[2] The growing cost of energy is one of several factors that have put green high-performance computing in the spotlight.

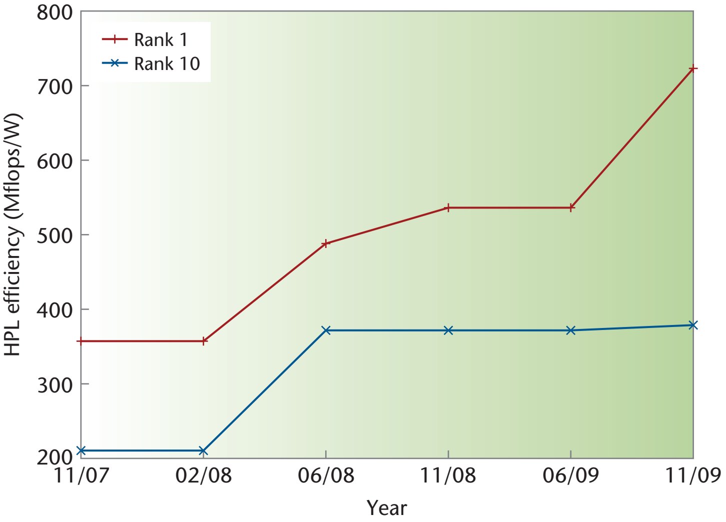

Figure 2. High-performance Linpack (HPL) efficiency for the Green500 list's rank 1 and rank 10 systems. The list's total number of cores increased from 1.2 million to 4.6 million, clearly showing the emerging importance of power-aware large-scale networking.

Measuring Energy Efficiency

Well-known energy-centric metrics—such as floating point operations per Joule (flops/J) or flops per second per watt (flops/W), can extend or even replace the current dominant measure of floating point operations per second (flops). Part of this movement is reflected in the Green500 list and other activities related to energy-efficient computing at extreme scales.

Figure 2 shows the development of the high-performance Linpack (HPL) benchmark's power efficiency on the Green500 list's first and tenth systems over a two-year period. During that time, the total number of cores in the Top500 list increased from 1.2 million to 4.6 million, which clearly shows the emerging importance of power-aware large-scale networking. Thus, we argue that such computation-centric metrics (flops/J) should be extended to capture the efficiency of data-movement such as bytes per joule (B/J).

The Interconnect's Role

Interconnection networks satisfy remote data-dependencies in parallel applications. For example, a dot product of two distributed vectors must ultimately return the global sum of all partial results, which obviously requires the exchange of the partial results over the network. Although this communication doesn't advance the actual

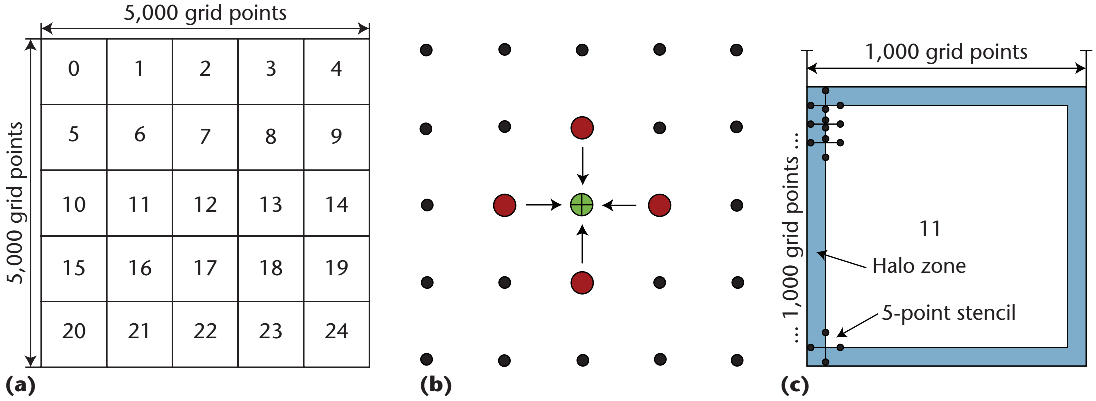

Figure 3. A Poisson solver example. The (a) domain decomposition, (b) five-point stencil, and the (c) halo zones, which consist of data that must be communicated.

computation, we need it to retrieve the correct results. Reducing *communication overhead*—that is, the time that an application spends in communication routines—is an important goal. To achieve it, most programmers strive to use the network as little as possible, keeping the ratio of communication to computation in a parallel program as low as possible.

We'll use a 2D Poisson solver to demonstrate typical communication overheads. Programmers use Poisson solvers to solve a large class of elliptic partial differential equations (PDEs) that model many physical phenomena, such as heat dissipation. Our example problem spans 5,000 grid points in each dimension, with 25 million grid points total.

We ran the computation on 25 processes arranged in a 2D 5 × 5 grid (see Figure 3a). We distributed the grid to the five processes, with each process receiving a 1,000 × 1,000 patch. In a standard five-point stencil, each grid point needs the values of four adjacent points in its computation (see Figure 3b) and requires communication of all (4,000) elements at the process boundary to the adjacent processes. The data that must be communicated is called the *halo zone*; the halo zone is sent to a specialized communication buffer (the *ghost cells*). Figure 3c shows communication boundaries for process 11. Such solvers are usually run iteratively until convergence criteria are met.

It's possible to implement this algorithm in parallel by looping over the local grid and applying the stencil operation. When the loop arrives at a grid point that requires remote data, it can request this data from a neighbor (which, given the symmetry, can proactively push the data to the receiver). However, this very fine-grain communication is inefficient.

A simple way to reduce network overheads is to avoid small message transfers by aggregating messages into local buffers before sending them. In our example, the communication of *all* ghost cells would have to happen at the beginning of an iteration instead of when they're actually needed. This avoids the startup costs per message and thus reduces the overhead. However, implementations typically require local buffering and additional memory copies at the sender or receiver.

Data movement in dynamic RAM (DRAM) is one of the most energy-consuming operations and should be avoided. Thus, even node-local strategies, such as message aggregation, should be used with care. One possible optimization would be to pass blocks of data without buffering them to the communication layer. The message passing interface (MPI) offers this possibility, using derived data types to let users send data that's scattered through memory without packing it in a contiguous buffer.

Energy Saving Strategies

Most of today's interconnection networks have no means to save power; they can't, for example, turn off links or clock down circuits. Thus, even when idle, network links run at full steam, typically performing physical link-level control protocols. Bursty communication requires relatively high bandwidth (that is, wide links and high frequency) and thus high power consumption. But, because most programmers strive to keep communication overhead low, they try to keep the network largely idle.

In this situation, there are two obvious ways to save power: enable power-saving mechanisms in the network, and alter algorithms and applications to use the network more efficiently.

Hardware Power Savings

Industry groups have analyzed power-saving techniques for network interconnects. IEEE's

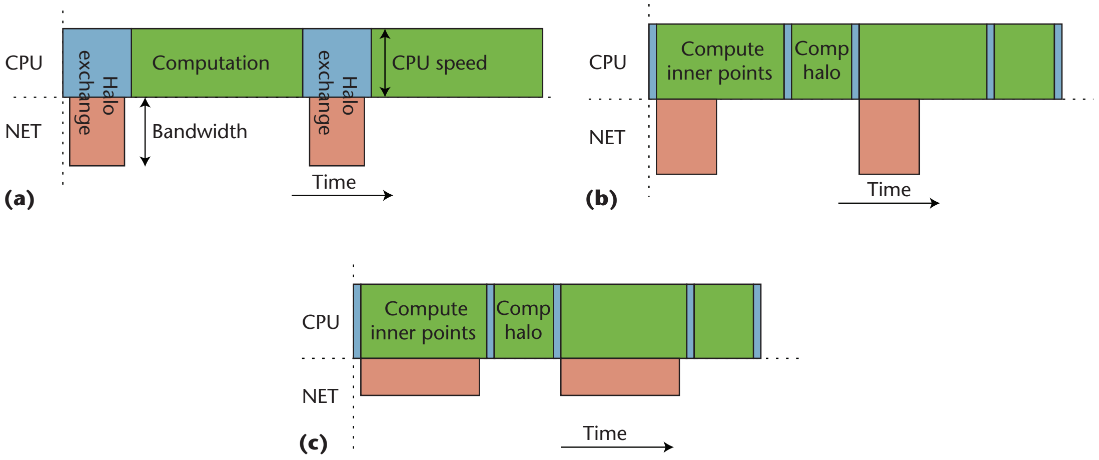

Figure 4. Poisson example with power saving optimizations. Power-saving possibilities include (a) blocking communication, (b) nonblocking communication with overlap, and (c) overlap with half bandwidth.

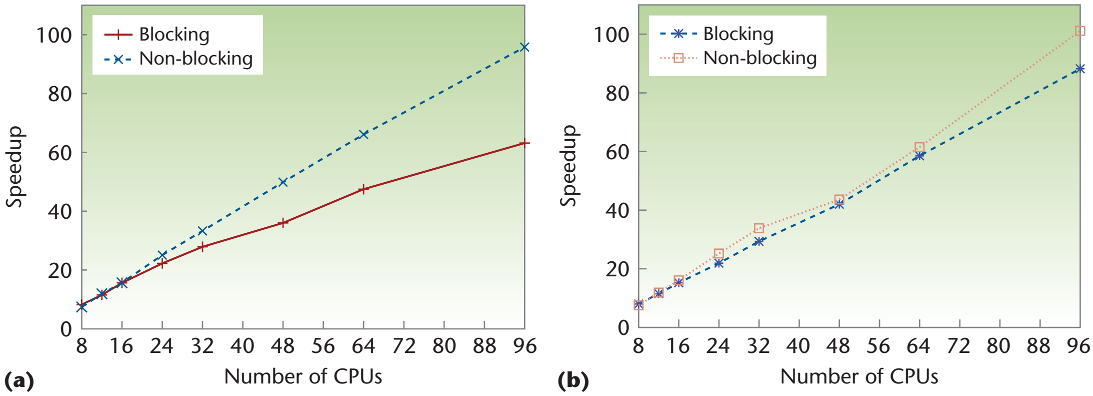

Figure 5. Parallel speedup for (a) Ethernet and (b) InfiniBand. Optimizing through overlapping computation and communication produced equally good performance in both networks. The applied optimizations avoid memory copies and bursty network traffic and thus reduce the power consumption.

task force on energy efficient Ethernet (IEEE 802.3az) discussed power saving options such as dynamic link-speed reduction, receiver modification, and deep sleep states. However, some of those options might hinder today's high-performance applications because state-changes from sleep to active could take up to 10 microseconds, which directly adds to the latency and significantly increases network jitter. In general, while such mechanisms provide a huge power-saving potential, applications should be notified if performance is degraded.

Algorithmic Power Savings

Unlike hardware, which is typically replaced in four-year cycles, algorithmic changes are usually more difficult to implement because algorithms and parallel codes are often significant long-term investments. However, initial results show that overlapping communication and computation can lead to a steady use of the interconnect, which in turn reduces the required bandwidth and/or improves application performance and wait times. Such optimizations have two effects. First, they reduce the time to solution (see Figure 4b). Second, they allow systems to throttle network speeds and avoid bursty traffic (see Figure 4c).

In a previous study,[8] we analyzed the effects of overlapping computation and communication over Gigabit Ethernet and SDR InfiniBand. Both networks' effective bandwidth differs by a factor of 8, and the initial costs differ by nearly a factor of 15. Our Poisson solver with 800^3 points on 96 nodes performs 34 percent worse over Gigabit Ethernet than over InfiniBand. However, as Figure 5 shows, with overlapping computation and communication, both networks perform equally well. The applied optimizations avoid memory copies and bursty network traffic and thus reduce the power consumption.

Optimized problem partitioning and task mapping could also save power by improving locality.[9] However, it's important to weigh the overheads of computing the partition and redistributing the data against the savings in memory and network usage.

Not all algorithms and applications readily lend themselves to overlap like the Poisson solver. Often, other more complex techniques must be employed to extract the needed parallelism of communication and computation. Among the promising techniques are one-sided communication (such as MPI-2 One-Sided) and partitioned global address space models (such as Unified Parallel C or Co-array Fortran). These eliminate the receiver's CPU overhead (with appropriate hardware support) and thus enable efficient fine-grained communication. This alleviates the need for coalescing or at least takes it out of the application developer's hands. Power-optimized libraries could use the relaxed memory consistency of remote memory access models (such as MPI-2) to transmit messages most efficiently, and programmers could overlap fine-grained communication and computation conveniently.

Not all algorithms and applications readily lend themselves to overlap like the Poisson solver. Often, other more complex techniques must be employed to extract the needed parallelism of communication and computation.

Analysis: Small-Scale Network Power Consumption

Future developments in software and hardware are on their way, but several limitations apply to today's systems. Here, we report gathered data comparing the power consumption of two different networking technologies under microbenchmark conditions, as well as full application runs.[10]

Our study compares the Mellanox double data rate (DDR) InfiniBand ConnectX (MT26418) with Myricom's Myrinet 10G on 14 dual symmetric multiprocessors (SMPs), quad-core IBM iDataPlex (dx360) nodes. We used identical hardware and software and simply swapped the cards and switches for the benchmarks.[10] Our InfiniBand setup was based on copper (configuration A) and we compared copper-based (10G-PCIE-8A-C, configuration B) and fiber-based (10G-PCIE-8B-QP, configuration C) Myricom cards.

The complete system's power draw at 120 volts varied between 17.7 and 40 amps idle and under maximum load, respectively. Interestingly, the idle power was 17.7 amps for configuration A, 17.3 amps for configuration B, and 16.9 amps for configuration C, which is nearly a 5 percent difference. We also observed a similar difference under high communication load, which increased the power consumption by more than 20 percent. Because the networking hardware doesn't yet support power savings, the increase in total power consumption is likely due to the CPU's higher use of the endpoints.

Our study used four applications:

- Multiple Instruction, Multiple Data (MIMD) Lattice Computation (MILC/su3 rmd),
- Parallel Ocean Program (POP),
- Wave Propagation Program (WPP), and
- Random Axellerated Maximum Likelihood (RAxML).

All programs use MPI for communication and exhibited between 4 and 27 percent communication overhead.

We recorded the power consumption for complete runs of all four applications; Figure 6 shows the power traces. Based on this data, we computed the total energy consumption for each configuration (see Table 1).

It would be interesting to compute the flops/J for each application as is often done for the HPL benchmark. A next step could then be to assess each application's power efficiency with regards to some theoretical upper limit. To determine the floating point efficiency, researchers typically assess the application's efficiency by comparing the reached flops with the peak flops. Because this model neglects other system parts—such as the memory subsystem—it's questionable in itself. However, such metrics do encourage users to make more efficient use of other resources, such as memory or network.

Determining a *power-efficiency measure* is harder because deriving an upper bound—that is, determining the flops/J for an ideal application—is a nontrivial task in itself and probably requires a vendor specification (such as the theoretical peak flops rate). This and equivalent iso-energy-efficiency scaling (such as isoefficient scaling) is an interesting topic for future research.

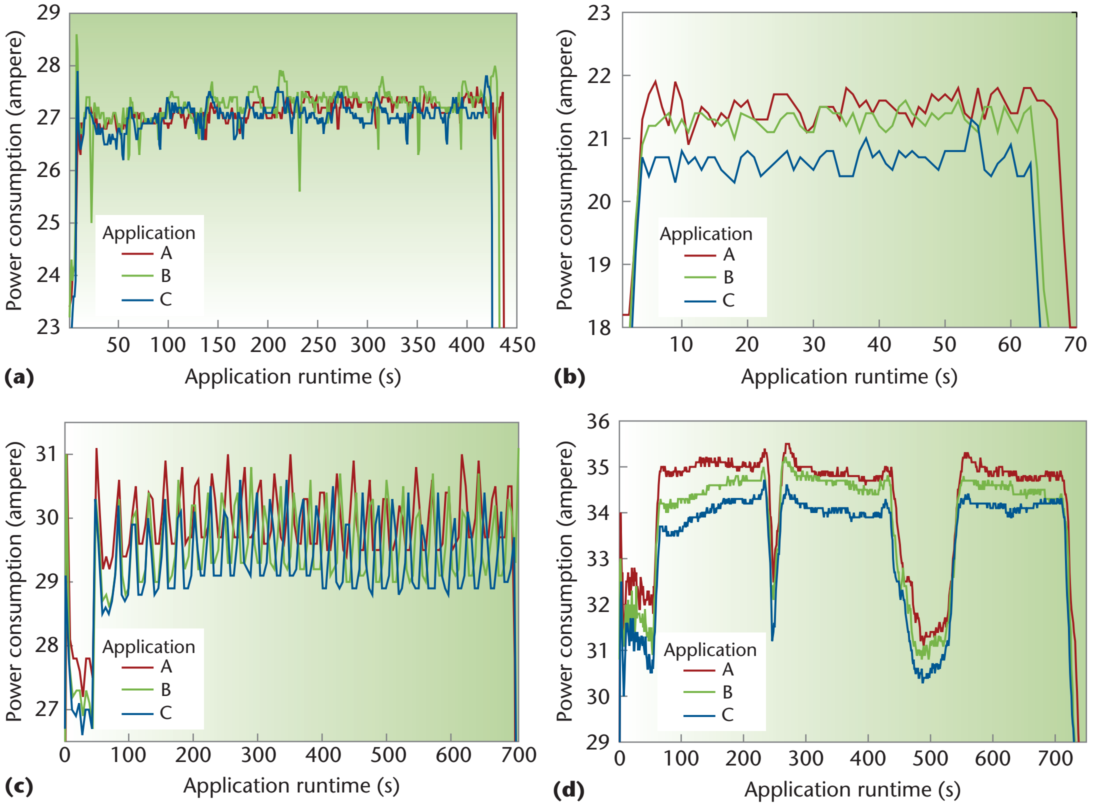

Figure 6. Power-profiles for all configurations and networks. (a) Multiple Instruction, Multiple Data (MIMD) Lattice Computation (MILC), (b) Parallel Ocean Program (POP), (c) Wave Propagation Program (WPP), and (d) Random Axellerated Maximum Likelihood (RAxML).

Table 1. Total power consumption (in kilowatts per hour) for each problem set.

Application	Configuration A	Configuration B	Configuration C
MIMD Lattice Computation (MILC)	3.879 (100%)	3.875 (99.9%)	3.819 (98.5%)
Parallel Ocean Program (POP)	0.458 (100%)	0.432 (94.3%)	0.406 (88.6%)
Wave Propagation Program (WPP)	6.807 (100%)	6.781 (99.6%)	6.713 (98.6%)
Random Axellerated Maximum Likelihood (RAxML)	8.315 (100%)	8.164 (98.2%)	8.015 (96.4%)

Toward Large-Scale Systems

Scaling interconnection networks is hard and requires several trade-offs. These trade-offs are most important when it comes to choosing the network topology—that is, the way in which the networking elements are connected.

The network topology determines the two most important characteristics of an interconnection network: bisection width and diameter. The bisection width defines the minimum number of links (cables) that must be removed to split the network into two equal parts. It also determines the performance of applications that communicate large messages in dense communication patterns (such as all-to-all or random patterns). The network diameter mainly influences the performance of applications that communicate small data, such as scalar values.

Low-diameter networks often have a high bisection width, so the basic trade-off is cost/power consumption versus diameter and bisection width. Many scientific applications require only sparse—or even regular—communication patterns, such as 2D and 3D Cartesian grids as in our Poisson example. Developers can often efficiently embed such applications in low-cost topologies, such as torus networks, where the number of switches scales favorably ($O(P)$) with the network size (P). However, achieving efficient parallel-application embedding remains a topic of ongoing research, and most implementations support only the embedding of Cartesian grids

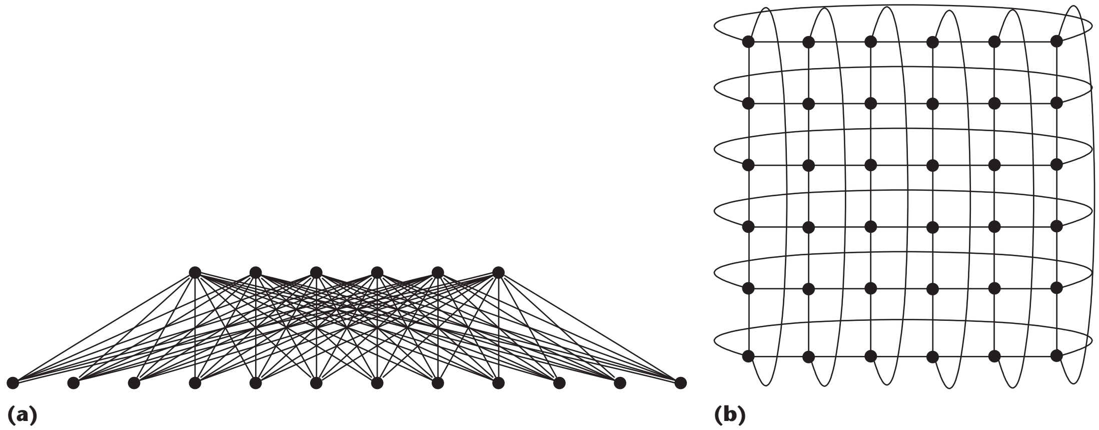

Figure 7. Torus and fat-tree network topologies. (a) The two-stage fat-tree topology has 144 nodes (24-port switches, with 12 nodes per level-1 switch). (b) The 2D torus topology has 36 nodes (four-port switches, with one node per switch).

into torus topologies.[11] Most large-scale systems either have deep hierarchies (such as Road Runner) or torus networks (such as Blue Gene or Cray XT systems) as topologies.

Some communication patterns, such as wheel or star graphs, are hard to embed into torus topologies and many applications require the full bisection width and/or low diameter. Examples of such applications are 3D fast Fourier transformations (FFTs), which perform large-data transpose (all-to-all) operations, or parallel graph computations with data-driven and highly dynamic communication patterns.

Example Topologies

To support all applications that require maximum bisection width and minimal diameter, we need networks in which the number of required switches grows superlinearly ($O(P \log P)$), such as Clos or fat-tree networks. However, the power consumption of such networks grows significantly at scale. We'll again do a simple calculation based on our measurement results. InfiniBand switches (configuration A) have 24 ports while Myrinet switches (configuration B and C) have 32 ports. Both switches draw 0.48 amps (This is a pure model computation; different integration and port counts will change the computation and shift some constants in the results.)

In the following, we'll extrapolate our simple data to a large-scale system as described in the US Department of Energy's report on exascale systems.[12] Our envisioned exascale scenario consists of 1.3 million processing elements with 64 cores each. If we assume SMP systems with 8 processing elements (PEs) per node, our model system would have 162,500 network endpoints. The network would have three hierarchies:

- the on-chip network connecting all cores,
- the off-chip network connecting all PEs in a node, and
- the off-node network, which we discuss here.

We compare two typical network topologies. The torus topology scales with O(P) switches linearly with the number of nodes, but has a high diameter and low bisection width. The fat-tree topology that scales with O(P log(P)) switches superlinearly with the number of nodes and has low diameter and high bisection width. Figure 7 shows a 2D torus with 36 nodes (four-port switches, with one node per switch) and a two-stage fat-tree with 144 nodes (24-port switches, with 12 nodes per level-1 switch).

In the 3D torus network, each node has six neighbors. If we built a torus with full bandwidth to all neighbors, we could attach three nodes to each InfiniBand switch and four nodes to each Myrinet switch. This would leave three and four ports per switch idle, respectively. If we scaled this network to 162,500 nodes, we would need 54,167 InfiniBand or 40,625 Myrinet switches. In the fat-tree network with full bisection width and a similar number of endpoints, we would need a five-level fat-tree requiring 60,939 and 45,706 switches for InfiniBand and Myrinet, respectively.

In both cases, the interconnect would represent a significant fraction of the total power consumption; the switches alone consume as much power as several thousand computer nodes under full load. However, integration and more power-effective designs will likely lower this consumption significantly. IBM's Blue Gene supercomputer, for example, integrates the torus network in the CPU chip.[13] Nonetheless, the network transceivers that drive the external network links use a significant fraction of the switch's power.

Routing and Fiber-Based Technologies

Network routing should also be investigated for power savings. For example, destination-based distributed routing as used in InfiniBand or Ethernet networks requires a look-up in a forwarding table at each switch. Source-based routing as used in Myrinet or Blue Gene simply encodes the route in each packet's header, enabling a much simpler logic in switches. For the latter, it's necessary to include the power costs of encoding the route into each packet. There's thus no general silver bullet to save energy in routing and topology of large-scale networks; most decisions require careful analysis of all alternatives in the limits of current technology.

An interesting emerging technology is the wide availability of fiber-based interconnects. While the first generation of these interconnects uses the same electrical connectors and are thus not saving much energy, it's possible to drive fiber cables with less energy over higher distances than copper-based cables. The higher cable lengths permit the construction of networks with lower diameters and full bisection width, such as the flattened butterfly.[14] This again contributes to energy savings and performance improvements. However, today's optical networking is relatively unoptimized. This is due both to low market volume and to the expense of crossing over from copper to optics—in the range of several meters for cost and energy consumption.

It's now up to the hardware and software communities to make network energy consumption an explicit variable in the equation. Methods, such as power saving states and low-power optics, could lead to quick successes on the hardware side. Power savings in software are at least as important and have a huge potential. Software development tools and languages can help programmers enable low-power applications that use the networking resources more effectively.

Toward exascale, we need new system designs, topologies, and programming environments to stay within the practical energy limitations. All such developments must consider energy consumption hand-in-hand with the performance requirements.

References

1. L.A. Barroso, "The Price of Performance," *ACM Queue*, vol. 3, no. 7, 2005, pp. 48–53.
2. *Monthly Energy Review*, Oct. 2008, US Energy Information Admin., 2008; www.eia.gov/FTPROOT/monthlyhistory.htm.
3. D. Klein, *Challenges in Energy-Efficient Memory Architecture*, presentation, Power Efficiency and the Path to Exascale Computing Workshop, 2008; www.lbl.gov/CS/html/SC08ExascalePowerWorkshop/Klein.pdf.
4. M. Taiji, "MDGrape-3 Chip: A 165-Gflops Application-Specific LSI for Molecular Dynamics Simulations," *Proc. 16th IEEE Hot Chips Symp.*, IEEE CS Press, 2004, pp. 49–63; doi:10.1145/1188455.1188506.
5. D.E. Shaw et al., "Anton, a Special-Purpose Machine for Molecular Dynamics Simulation," *Computer Architecture News*, vol. 35, no. 2, 2007, pp. 1–12.
6. W.C. Feng, "Making a Case for Efficient Supercomputing," *ACM Queue*, vol. 1, no. 7, 2003, pp. 54–64.
7. N.R. Adiga et al., "An Overview of the Blue Gene/L Supercomputer," *Proc. Int'l Conf. High-Performance Computing, Networking, Storage, and Analysis*, ACM Press, 2002, pp. 1–22.
8. T. Hoefler et al., "Optimizing a Conjugate Gradient Solver with Non-Blocking Collective Operations," *J. Parallel Computing*, vol. 33, no. 9, 2007, pp. 624–633.
9. P. Raghavan, *Energy-Aware Algorithms at Exascale*, presentation, Power Efficiency and the Path to Exascale Computing Workshop, 2008; www.lbl.gov/CS/html/SC08ExascalePowerWorkshop/PRSCfinal.pdf.
10. T. Hoefler, T. Schneider, and A. Lumsdaine, "A Power-Aware, Application-Based, Performance Study of Modern Commodity Cluster Interconnection Networks," *Proc. 23rd IEEE Int'l Parallel & Distributed Processing Symp.*, IEEE Press, 2009, pp. 1–7.
11. H. Yu, I.H. Chung, and J.E. Moreira, "Topology Mapping for Blue Gene/L Supercomputer," *Proc. Int'l Conf. High-Performance Computing, Networking, Storage, and Analysis*, IEEE CS Press, 2006, no. 116; http://doi.acm.org/10.1145/1188455.1188576.
12. H. Simon, T. Zacharia, and R. Stevens, *Modeling and Simulation at the Exascale for Energy and the Environment*, E3 report, US Dept. Energy, 2007; www.sc.doe.gov/ascr/ProgramDocuments/Docs/TownHall.pdf.
13. M. Blumrich et al., *Design and Analysis of the Blue Gene/L Torus Interconnection Network*, report, IBM, 2003.
14. J. Kim, W.J. Dally, and D. Abts, "Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks," *Proc. 34th Int'l Symp. Computer Architecture*, ACM Press, 2007, pp. 126–137.

***Torsten Hoefler** leads the Blue Waters project's performance modeling and simulation efforts at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. His research interests focus on large-scale HPC networking and parallel programming frameworks. He a coauthor of MPI-2.2, chair of MPI-3's collective operations working group. Hoefler has a PhD in computer science from Indiana University. He is a member of IEEE and the ACM. Contact him at htor@illinois.edu.*