

The Convergence of Hyperscale Datacenter and High-performance Computing Networks

Torsten Hoefler
ETH Zürich

Ariel Hendel
Scala Computing

Duncan Roweth
Hewlett Packard Enterprise

Abstract—The emergence of large-scale distributed data processing and complex datacenter services led to an increase of intra-datacenter traffic. The characteristics of this emerging traffic are similar to those in high-performance supercomputers. However, network technologies used in supercomputers and datacenters are quite different and it is natural to ask whether they could be unified. We discuss the differences and commonalities between these two workload types and technologies and outline a path to convergence at multiple layers. We predict that emerging smart networking solutions will accelerate that convergence.

■ INTRODUCTION

In recent years, datacenter computing has experienced an unprecedented growth leading from in-house server rooms to mega-, hyper-, and warehouse-scale datacenters. The number of network endpoints in these systems has passed the size of the world's largest supercomputers, which themselves just reached the Exascale frontier. The network of first-generation datacenters was mostly serving data to external clients and supported simple distributed applications running in

the datacenter. However, with the advent of large-data processing and machine-learning, datacenter network requirements quickly incorporate aspects of traditional high-performance computing. *These new traffic demands started a discussion whether high-performance and traditional datacenter networking should converge.* While the resulting economy of scale is attractive, there are also several aspects hindering convergence. In this paper, we point out differences and commonalities between high-performance and datacenter com-

puting and their impact on the technological developments in large-scale networking. We come to the conclusion that *smart high-performance datacenter networks enabling both High Performance Computing (HPC) and Mega Data Center (MDC) workloads will be adopted in industry soon.*

HPC has always pushed the limits of computing. The top-class systems, called *supercomputers*, have the highest concentrated compute capacity on earth. While most supercomputers are running multiple applications simultaneously, they are designed to run a single “hero run” application on the full machine to solve the world’s most challenging problems such as finding vaccines amidst a pandemic or training the largest deep learning models. Within today’s technology constraints, supercomputers are no longer built as a single server but consist of tens of thousands of separate servers connected by a high-speed communication network. The network (aka., *interconnect*) is the most critical component and supercomputer designs are centered around particular network architectures. This makes the network a major distinguishing factor because the “single application” scenario often has stringent latency and bandwidth requirements. One can say that it is *the interconnect that turns a set of servers into a supercomputer.*

HPC systems run *parallel applications* that are most commonly implemented on distributed memory supercomputers using the Message Passing Interface (MPI, [1]). MPI programs run similar code as processes on each server and algorithms are most often designed using the bulk synchronous parallel (BSP) computation model as a series of compute-communication-synchronization phases. Here, the application can only progress to the next phase once all processes finish synchronization. This has later been (re)discovered in MDCs as the problem with long tails [2]. Numerous programming techniques can reduce synchronization and communication overheads (e.g., [3], [4]), however, BSP applications are limited by latencies in extreme scaling scenarios. In fact, the communication latency (tail) distribution determines the system’s scalability limits, and determines the maximum number of processes a single application efficiently use [5].

Supercomputers have been overtaken in size

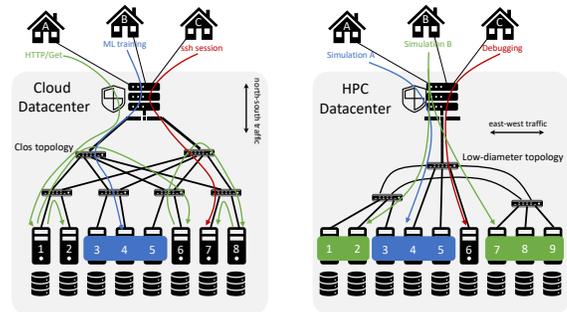


Figure 1. Usage scenarios of datacenter and HPC machines. The cloud datacenter serves multiple customers with different interactive services, some of them distributed (as the ML training job on machines 3-5). The HPC datacenter on the right primarily serves three distributed simulation workloads where clients are not waiting for an immediate answer.

by warehouse-scale mega-datacenters. The modern networked world creates the need to store and process the data consumed by connected client devices. Each of us now has multiple such mobile devices and is generating and consuming an ever-increasing amount of cloud-centric computation and storage. Additionally, not all client endpoints have necessarily humans consuming data or services behind a device. As the Internet of Things (IoT) gains popularity, hundreds of millions of devices stream data such as images, videos, and webpages to and from datacenters worldwide. MDC scale at AWS, Google, Facebook, or Microsoft is larger than the largest single HPC system and they run many more diverse simultaneous applications on the same compute, storage, and network infrastructure in support of a larger number of interactive end users. The scope of an MDC operator is its global user base, growing with adoption while the scope of an HPC operator is a well-defined application capacity planned at inception time. Figure 1 shows a sketch of HPC and MDC workloads.

MDC systems run *distributed applications* in which asynchronous processes communicate pairwise using programming interfaces such as Remote Procedure Calls (RPC). These applications rarely require the use of multi-server or global synchronization, therefore reducing the influence of latency on aggregate application performance. Increased latency on a single communication

between endpoint pairs only affects individual requests and not the whole application. Whenever many-to-one communication patterns emerge in MDC applications, for example *incast* patterns in map-reduce or distributed file systems, developers typically rely on soft-deadlines to mitigate the impact of the long tail in response latencies. The resulting applications do not stall under unbounded tail latency but rather compromise on result quality or efficiency. This is achieved by simply ignoring late RPC responses or launching them redundantly on different servers. Thus, network deficiencies do not slow applications but instead result in wasted capacity (which can be recovered by adding more servers).

The traditional role of datacenters was to store, process, and deliver the data to end-customers driving so-called *north-south* traffic from its servers to the Internet. When the Internet-facing path was the bottleneck, the datacenter network capacity could be relatively modest. However, in today's era of distributed data analytics and machine learning, the throughput and latency requirements for interconnection networks have grown steadily and the *east-west* traffic associated with communications between servers dominates by orders of magnitude. In that sense MDC traffic resembles traditional HPC applications, albeit with a more latency-forgiving model. For some emerging applications it becomes apparent that HPC and modern *big data analytics* (e.g., deep learning, document search, or recommendation systems) have similar computation and communication patterns. For example, much of machine learning can be expressed as tensor algebra, and collaborative filtering resembles traditional graph analytics on bipartite graphs. The major differences between those big data workloads and traditional HPC workloads is that the former emphasizes programmer productivity, while the latter emphasizes performance. Programming environments may continue to evolve along different paths for various reasons but we argue that the underlying workloads and their computational characteristics are very similar and converging quickly.

Yet, these workloads are executed using very different interconnects: HPC networks are optimized for highest performance while MDC networks follow traditional datacenter deployment

and operational philosophies. When looking into the details, it becomes apparent that the lowest levels have already converged, and commonalities exist as we move up the stack. Furthermore, the addition of high-performance accelerators (e.g., General Purpose Graphics Processing Units, GPUs) with higher bandwidth demands necessitates specialized networking in today's MDCs that lead to islands of HPC-like connectivity. These systems often supplement the *frontend* datacenter network with a specialized HPC-like *backend* network. Examples include Google TPU's dedicated torus interconnect and Azure HPC's InfiniBand deployments connecting GPU servers. This duplication leads to significant inefficiencies—given that the lower layers are already identical and just the communication protocols differ! In fact, endpoint solutions such as AWS Nitro and Microsoft's Catapult [6] attempt to optimize existing Ethernet networks. From the other side, Cray's Slingshot technology [7] comes from an HPC-centric view and adds Ethernet compatibility. These examples show how requirements and solutions are suggesting a common high-performance networking solution.

While the networking requirements of HPC and MDC are similar at a high level, the devil lies in the details. We now discuss a series of requirements where HPC and datacenter networking differs, ranging from design and deployment philosophy to application programming interfaces. We comment on how fundamental each of these differences is and hint that in-network compute solutions based on smart network interface cards (NICs) and switches will bridge many of those differences in the future. We conclude each section with a brief technology prediction.

Design and deployment philosophy

The most significant difference between the two networking viewpoints is the way machines are deployed. An MDC is naturally a loosely connected set of servers from multiple vendors that is incrementally expanded and upgraded. The cabling infrastructure lives through multiple generations of machines and technologies. MDCs install fiber as a building infrastructure and thus decouple the infrastructure and large parts of the network topology from the servers. The rack switch represents the architectural boundary

between the datacenter network and the compute servers. Multi-vendor support is fundamental and builds on Ethernet for the physical layers and the internet protocol (IP) for the higher levels. Speed heterogeneity is also fundamental to MDC networks where different servers may be attached at different link speeds, and the interior network links may be different from the endpoint speeds. MDC operators cannot afford extensive downtime for reconfiguration and must run multiple technologies at the same time. This incremental upgrade in MDCs makes modernization challenging and prohibits big jumps in technology.

Supercomputers are traditionally seen as one-off installations and are often designed and cabled as such: link speeds are identical for all endpoints and for interior links; their networks use components from a single vendor; plans for upgrades are usually made before the initial installation. Given the importance and cost of high-bandwidth interconnects, many supercomputers go beyond Clos networks or fat trees as interconnect topologies. Designs range from hypercubes or high-dimensional torus networks [8] to more cost-effective low-diameter topologies [9], [10]. Their deployment model allows supercomputers to adopt radical changes to network technologies with each new generation of the system. HPC sites run old and new systems in parallel, migrating the workload before decommissioning systems. This model of operation is expensive in floor area, power, and cost, and HPC operators are pressing for a more incremental approach.

 The incremental deployment and backwards compatibility requirements hinder the adoption of many innovative technologies in MDCs. HPC systems will continue to spearhead completely new, revolutionary directions in technology.

Operations philosophy

Historically, datacenters and HPC centers took very different views of their operations. This was mandated by their clients: cloud datacenters serve end-customers ranging from users of mobile phones to banks and hospitals. They run I/O-heavy workloads as live services where outages are visible within seconds and can cause large

financial penalties. The data gathered, credit card transactions for example, cannot be reconstructed, and any losses are detrimental. Thus, the offered services must be highly reliable and always available. Supercomputers have developed along a different path, one that trades reliability for performance and cost, where small outages (a few hours per year) can be tolerated. Individual jobs can fail, provided they can be rerun within the time allowed by the service level agreement (SLA) and computational resources are over-provisioned to allow this. This enables HPC operators to adopt much more risky deployments in software and hardware and in general be much more aggressive than MDC operators in terms of network and hardware technologies.

MDC networks prioritize network availability by combining mechanisms to ensure partial operation (e.g., separate network planes for fault isolation) with distributed protocols for control plane redundancy. HPC interconnects use separate management networks for reliability but rely on a centralized control plane for the high-performance network accepting short periods of unavailability in favor of efficient management. Applications running on MDCs implement reliability using complex redundancy at the software level (e.g., using standby services on separate servers or replicated storage). Applications on failed endpoints are quickly restarted on new resources and connected back to the service. This enables operators to use less reliable and cheaper hardware at the cost of additional software overheads. HPC applications on the other hand rely on restarting applications from checkpoints after failures. To reduce restart cost at large scale, HPC vendors use more reliable hardware than MDCs, e.g., an HPC network protects communication using both link level and end-to-end retry. Thus, HPC software has low reliability overheads while MDCs have to employ expensive replication and consensus schemes. MDC network operators can learn from HPC, with more advanced hardware fault tolerance, e.g., use of link-level retry.

Security is an important consideration for any computing system. HPC systems have traditionally less stringent requirements for software [11] and hardware security and often rely on physical security (e.g., air gapped systems and building protection) and avoid multi-tenancy on nodes.

The system administrator is a trusted entity and users are admitted carefully to systems. MDC systems serve sensitive third-party workloads and their tenants do not trust the operator or other tenants, who could literally be anybody with a credit card. This necessitates much higher levels of security in MDCs and motivates the emergence of solutions such as trusted execution or general confidential computing as well as secure high-performance networking [12]. Recently, a growing number of HPC systems hosts sensitive data (e.g., medical records) in shared file systems necessitating the need for MDC-like security concepts.

MDCs are operated by a remarkably low number of personnel; their scale is so large that it is not practical to have a human-based model for operations and automation is a must. This mandates a sophisticated monitoring, logging, and control infrastructure that is not present in HPC systems. Monitoring is key to both troubleshooting and capacity management. We have not discussed capacity, but “workload anxiety” is an important factor in MDC network design. It arises from the fact that the compute and storage capacity must be provisioned to absorb unpredictable changes in end user traffic and application workload profiles. The network must tolerate such compute, storage, and workload variations without a major redesign.

MDCs loathe considering physical affinity when deploying or provisioning apps, because capacity is deployed chronologically, and affinity would complicate virtual machine (VM) allocation strategies. Furthermore, availability SLAs require that applications are distributed across datacenters within a region or availability zone. Locality is often considered in HPC application deployment. While local placement is relatively simple on recursively structured networks such as fat tree or Clos networks, it is harder to achieve on other topologies. Yet, full global bandwidth networks promise to make placement decisions less critical.

 The fundamental difference lies in the approach to (network) availability and security. The operations side of HPC and MDC networks will close the gap if HPC operators implement the more stringent requirements posed by MDC operations. Other aspects are more similar and will likely converge.

Service diversity

MDCs reflect the business model of their operators. An operator that sells VM capacity to corporate customers (e.g., Microsoft) has different network profiles, controls, and SLAs than a “end-user-centric” operator focusing on interactions between humans (e.g., Facebook). However, all MDC operators use virtualization and multi-tenancy to improve management and resource utilization. Virtualization impacts the network as it drives the use of overlays allowing traffic to be routed to virtual rather than physical endpoints. Current HPC interconnects have no such virtualization or multi-tenancy needs and minimize overheads using bare-metal addressing.

MDCs host a plethora of services with very different traffic demands. For example, throughput workloads such as backup traffic, replication, and storage will share the same physical links as latency-sensitive traffic such as distributed computations and client interactions. This poses stringent quality of service (QoS) requirements on MDC networks. HPC networks run parallel computation and file-I/O and QoS has not been a priority, although it is becoming more important as workload diversity increases. For example, all-reduce operations used in many HPC and AI applications perform well on a quiet network, but traffic of other tenants can impact scalability significantly [13]. It is interesting to note that the HPC interconnects used in the US Exascale systems provide both QoS and advanced congestion management.

The size of MDC networks is limited by the electrical power that can be reliably supplied, not by application scalability. Today’s MDC networks span multiple locations and regions to ensure availability in the face of massive faults. This introduces high inter-datacenter traffic, which is different from traditional intra datacenter east-

west traffic and client facing north-south traffic. HPC traffic, on the other hand, is dominated by local communication that stays inside a single datacenter.

 Services running on MDC networks will continue to require a wide range of QoS classes. HPC systems will see a growing service diversity, which will make MDC-style mechanisms relevant.

Protocol stacks and layers

The Open Systems Interconnect (OSI) layers specify a design pattern for communication protocol stacks ranging from the Physical Layer (L1) to the Application Layer (L7). The distinction between the layers is debatable but most Internet services can be mapped to them. The datacenter world has inherited much of the traditional Internet protocol stacks and only recently started moving towards more specialized protocols such as Datacenter TCP (DCTCP) or Datacenter Quantized Congestion Notification (DCQCN). HPC networking, however, was always tuned for highest performance and does not provision the many headers (one for each protocol level) needed for a full OSI stack. For example, the transport layer L3 rarely exists in HPC interconnects because the network is not intended to be routable. Figure 2 compares the OSI layers of MDC and HPC systems.

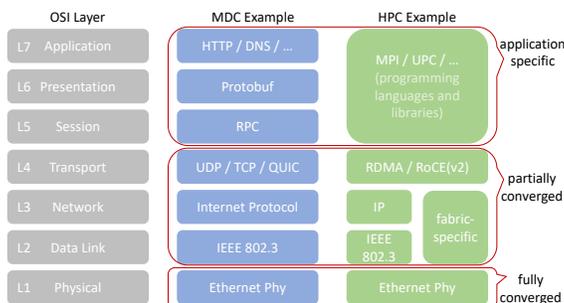


Figure 2. Open Systems Interconnect Layers.

At the electrical or optical signal level (L1), MDC and HPC networks are identical. Economy of scale in cabling and device infrastructure and the numerous technical constraints ensure that whoever gets there first is the winner. Ethernet has been winning this race for years with 25G, 56G,

and more recently 112G lanes. Some HPC and MDC networking technologies share L2-L4 but other HPC technologies employ proprietary protocols with more specialized and slimmer headers to achieve lowest overheads.

An interesting point of convergence is Remote Direct Memory Access (RDMA), which has long been used in HPC and storage networks to enable high-performance communication between the memories of a source and destination process at L4 or L5. The protocol is typically completely offloaded into a hardware implementation and operating system bypass reduces both latency and latency variance. Many MDC operators use or plan to use it in production (Azure, Google IRMA, AWS Nitro). At MDC scale however the sharing of buffers and bandwidth between RDMA and TCP/IP flows can victimize some traffic.

The simple hardware-based retransmit mechanisms in today’s RDMA network implementations rely on a lossless transport layer. However, most datacenter networks traditionally operate, like the Internet, with lossy routers that drop packets when queues are full. While the debate of lossy (endpoint-controlled flow rates) vs. lossless (network-controlled flow rates) has not been concluded, the requirement of lossless networking for RDMA raises a barrier to adoption in the conservative datacenter environment. For this reason, and to ensure lossless semantics, MDCs relegate RDMA traffic into dedicated QoS queues or physically separated in back-end networks.

 With growing link speeds, the relative bandwidth overheads of additional packet headers vanish and HPC networks may opt to support more complex routable protocols. We expect to see a move to message-based protocols over UDP/IP; RDMA over converged Ethernet (RoCE) is the first indication of this trend. Experimentation and optimization at MDC and HPC scales will be driven by discrete event network simulations such as distributed ns-3, SST, or LogGOPSim.

Network utilization

Utilization equals cost efficiency, one of the driving factors in both MDC and HPC systems. Because many MDC applications can tolerate

high latencies, their networks could in principle run at a high steady utilization and exceed 30-40% load on average without undue latency artifacts. However, the impact of dropped packets can be so detrimental that operators strive to keep utilization of network links well below the point at which packets start to be dropped.

Network utilization at network planning time is about estimating the end-to-end performance of all superimposed workloads. We found that applying network simulations to this phase enables an analysis of how “hot” the individual links can be operated at, what the switch buffer pressures are, and of course the level of packet drops and retransmits. Utilization at operation time is about monitoring the same links and switch buffers, and of course correlating drops and retransmits with links and buffers. Both simulations and operations can be SLA oriented, where then entire network utilization is sensed from latency distributions without much of a need to deal with bandwidth as a metric.

Large-scale BSP-style HPC applications operate in communication and computation phases, creating bursty on-off traffic patterns with stringent requirements on the latency distribution. HPC networks are engineered to meet the peak bandwidth requirements of bursty traffic. Utilization can be increased when the system is running multiple jobs, but contention between jobs, also known as the “noisy neighbor” problem, leads to critical latency variations. Performance isolation between applications can mitigate this problem and is thus a concern in both MDC and HPC networks. MDC operators enforce rate limiters at the traffic source (typically a VM) to address network performance isolation. In HPC, ensuring minimizing performance variability requires limiting the interaction between applications and their traffic types because both *system noise* [5], [14] and *network noise* [13], [15] have a detrimental impact on application performance. The single vendor model used in HPC networks has allowed deployment of novel hardware congestion management mechanisms (e.g., [7]) that operate at much finer granularity.

Static equal cost multipathing (ECMP) can create congestion hotspots, especially with a small number of communication-intensive flows. Adaptive routing or packet spraying can increase

network utilization while controlling the danger of transient packet drops. However, up until recently most merchant Ethernet switches did not offer adaptive routing or packet spraying because MDC network endpoints do not support out-of-order packet reception well. Recently, adaptive flowlet routing, which promises to not change packet ordering while providing some limited form of adaptive path selection, was introduced in MDC switches. Adaptive routing is a prerequisite for the efficient use of low-diameter topologies (common in HPC), essentially allowing the simultaneous use of minimal and non-minimal paths. HPC network endpoints support out-of-order delivery using the RDMA transport where packets carry destination addresses and can be written to memory independently.



The rise of message-based protocols over UDP/IP that relax ordering requirements at the endpoints will enable routing approaches that go beyond static multipathing. We also predict fast-paced evolution in the congestion avoidance aspects of these transports, and of TCP itself.

Application and programming model requirements

Application requirements are shifting on both sides and seemingly converge in the middle. HPC used to be very low-level where programs run on the bare metal and access the network through a slim message passing (MPI) [1], [16] or remote memory access (RMA) [17] interface. These interfaces can offer overheads of 100ns or less to reach the wire and sub-microsecond end-to-end. MDC applications are typically relying on sockets with expensive copy semantics. Fast RPC frameworks [18] can potentially bridge the gap and enable transparent zero-copy in MDC environments.

Task-based HPC programming models use and extend these established interfaces to relax BSP’s latency requirements. Traditional MDC applications are relatively insensitive to latencies but emerging workloads, for example, new data analytics and deep learning workloads resemble BSP-style HPC applications and have similarly stringent latency requirements. However, programmer productivity, rapid prototyping,

and quick deployment play a more important role than performance in MDCs. Only mature applications and stacks are explicitly tuned for performance. Many applications are written in managed languages such as Java or Python and run in virtualized environments with up to 10 microseconds just to get to the network. HPC and MDC optimize at different levels: with HPC focusing on best use of CPU and networking resources and MDC focusing on productivity and utilization of the system as a whole.

The different application requirements lead to different network APIs. The transition to RDMA networking occurred nearly two decades ago in HPC which since then has operated with single-digit microsecond latencies. RDMA allows most of the communication work to be offloaded to the network interface. Virtual memory mechanisms allow the data path to bypass the host operating system and move data directly between endpoint memories. HPC programming frameworks expose the remote memory access semantics to the applications directly to minimize overheads [17]. MDCs are only slowly realizing the potential of these technologies [19]. Adoption is slow because RDMA does not fit the traditional TCP/IP sockets model and layered routing well but specifications such as RoCEv2 and Priority Flow Control (PFC) enable L3 routing and moves RDMA to MDCs.

Modern HPC networks go far beyond RDMA, with the NICs performing message matching and collective operations, offloading these tasks from the CPU or GPU to improve overlap of computation and communications. Smart NIC applications in MDCs are typically for the benefit of the provider, ensuring isolation and not to improve tenant applications. Multi-tenancy in MDCs makes offloading user-level logic significantly more complex than in HPC, where the NIC is typically owned by a single application. Generic Smart NIC programming interfaces such as streaming Processing in the Network (SPIN [20]) promise a versatile acceleration strategy that can be described as *CUDA for the network*.

 RDMA is ubiquitous in HPC systems today while MDC operators are adopting RDMA for a larger share of their traffic. Furthermore, we expect to see significant developments in programmable network accelerators in both MDC and HPC networks moving beyond RDMA's simple deposit-to-memory semantics.

Conclusions and predictions

While datacenter providers are busy adjusting to RDMA and packet-level routing approaches, the research community is quickly moving on to versatile stream processing in the network with smart NICs and switches. New devices for network acceleration and marketing terms such as DPU, IPU, or NPU, are hyped and pushed into the market by various vendors.

Their current main deployments are Microsoft's catapult and AWS' Nitro NICs - both as infrastructure support. Their main use-cases are to improve security (tenant isolation), efficiency (encapsulation and encryption offload), and cost (specialization and in-house development) for multi-tenant hosts. HPC systems yet have to deploy smart NICs at scale. We predict that their roles will soon include more generic network processing and offload of application-specific protocols to specialized hardware.

Since the main difference between HPC and MDC lies in the upper layers of the protocol stack, smart NICs and in-network computation may unify both through use of application-specific protocols. We will see socket-based (TCP/IP or QUIC) applications and MPI applications on the same network and smart accelerated NICs (cf. [21]) will implement the protocol differences. Furthermore, application-specific protocols are an exciting opportunity for in-network acceleration in both the endpoints and the switches. We will see switch-based in-network computation such as reductions for deep learning workloads [22] enabling workload specialization at all levels.

The term "smart" in relation to network components such as NICs or switches needs a rigorous definition beyond the current marketing terminology. We propose to call a network interface smart if it allows stateful computation on

messages or flows. With such a clear definition one can reason about the behavior of such smart networks.

We conclude that while HPC and MDC are converging at the application level, their feature requirements are different enough to support two lines of development. The current ecosystem forms an interesting feedback loop where groundbreaking new technologies can be driven by and tested in a risk-accepting HPC environment. Yet, the mass market will remain Ethernet which slowly absorbs successful technologies developed in HPC. One recent example is the advent of RoCE. Both HPC and MDC could significantly reduce costs by using the same hardware infrastructure if it were configurable through uses of smart NICs and switches. Central to the Ethernet brand is the promise of interoperability, which can form a solid base for both HPC and MDC networks, yet, vendors supporting RDMA yet have to live up to that promise.

In summary, while we do not know which technology will dominate the mass market in 10-15 years, it is certainly going to be called Ethernet.

Attribution

Binocular, House, Shield, Router, Switch, Computer, Disks, based on icons from marfuah, Guilherme Furtado, scott desmond, Creative Stall, Muhajir ila Robbi, DinosoftLab, respectively, from thenounproject.com.

REFERENCES

1. Message Passing Interface Forum, *MPI: a message passing interface standard*. Technical Report, September 2012.
2. J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, pp. 74–80, 2013.
3. T. Hoefler, C. Siebert, and A. Lumsdaine, "Scalable Communication Protocols for Dynamic Sparse Data Exchange," in *Proceedings of the 2010 ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'10)*, pp. 159–168, ACM, Jan. 2010.
4. T. Hoefler, A. Lumsdaine, and W. Rehm, "Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI," in *Proceedings of the 2007 International Conference on High Performance Computing, Networking, Storage and Analysis, SC07*, IEEE Computer Society/ACM, Nov. 2007.
5. T. Hoefler, T. Schneider, and A. Lumsdaine, "Characterizing the Influence of System Noise on Large-Scale Applications by Simulation," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10)*, Nov. 2010.
6. A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A reconfigurable fabric for accelerating large-scale datacenter services," *SIGARCH Comput. Archit. News*, vol. 42, p. 13–24, jun 2014.
7. D. D. Sensi, S. D. Girolamo, K. H. McMahon, D. Roweth, and T. Hoefler, "An In-Depth Analysis of the Slingshot Interconnect," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC20)*, Nov. 2020.
8. Yuichiro Ajima, *High-dimensional Interconnect Technology for the K Computer and the Supercomputer Fugaku*. Fujitsu Technical Review, June 2020.
9. J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," *SIGARCH Comput. Archit. News*, vol. 36, p. 77–88, jun 2008.
10. G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. Hoefler, "Cost-Effective Diameter-Two Topologies: Analysis and Evaluation," ACM, Nov. 2015. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC15).
11. B. Rothenberger, K. Taranov, A. Perrig, and T. Hoefler, "ReDMark: Bypassing RDMA Security Mechanisms," in *Proceedings of the 2021 USENIX Security Symposium*, USENIX, 2021.
12. K. Taranov, B. Rothenberger, A. Perrig, and T. Hoefler, "sRDMA – Efficient NIC-based Authentication and Encryption for Remote Direct Memory Access," in *Proceedings of the 2020 USENIX Annual Technical Conference*, USENIX, Jul. 2020.
13. D. D. Sensi, S. D. Girolamo, and T. Hoefler, "Mitigating Network Noise on Dragonfly Networks through Application-Aware Routing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC19)*, Nov. 2019.
14. F. Petrini, D. J. Kerbyson, and S. Pakin, "The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of asc q," in *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, SC '03*, (New York, NY, USA), p. 55, Association for Computing Machinery, 2003.

15. T. Hoefler, T. Schneider, and A. Lumsdaine, "The Effect of Network Noise on Large-Scale Collective Communications," *Parallel Processing Letters (PPL)*, vol. 19, pp. 573–593, Aug. 2009.
16. G. Ciaccio and G. Chiola, "Gamma and mpi/gamma on gigabit ethernet," in *Proceedings of the 7th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, (Berlin, Heidelberg), p. 129–136, Springer-Verlag, 2000.
17. T. Hoefler, J. Dinan, R. Thakur, B. Barrett, P. Balaji, W. Gropp, and K. Underwood, "Remote Memory Access Programming in MPI-3," *ACM Transactions on Parallel Computing (TOPC)*, Jan. 2015. accepted for publication on Dec. 4th.
18. J. Soumagne, P. H. Carns, and R. B. Ross, "Advancing rpc for data services at exascale," *IEEE Data Eng. Bull.*, vol. 43, pp. 23–34, 2020.
19. L. Barroso, M. Marty, D. Patterson, and P. Ranganathan, "Attack of the killer microseconds," *Commun. ACM*, vol. 60, p. 48–54, mar 2017.
20. T. Hoefler, S. D. Girolamo, K. Taranov, R. E. Grant, and R. Brightwell, "sPIN: High-performance streaming Processing in the Network," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC17)*, Nov. 2017.
21. S. D. Girolamo, A. Kurth, A. Calotoiu, T. Benz, T. Schneider, J. Beránek, L. Benini, and T. Hoefler, "A RISC-V in-network accelerator for flexible high-performance low-power packet processing," in *Proceedings of the 48th Annual International Symposium on Computer Architecture (ISCA'21)*, Jun. 2021.
22. D. D. Sensi, S. D. Girolamo, S. Ashkboos, S. Li, and T. Hoefler, "Flare: Flexible In-Network Allreduce," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC21)*, ACM, Nov. 2021.

Torsten Hoefler is a Professor of Computer Science at ETH Zürich, Switzerland. He is a key member of the Message Passing Interface (MPI-3) Forum where he chairs the "Collective Operations and Topologies" working group. He has worked with high-performance networking for more than 15 years beginning with early InfiniBand systems and has played key roles in the analysis and deployment of various InfiniBand, IBM's PERCS, Cray's Aries and Slingshot networks, as well as several large-scale Ethernet installations. For more information see <http://htor.inf.ethz.ch>.

Ariel Hendel's interest is Data Center Networks,

guiding silicon innovation to maximize large-scale Data Center efficiencies. His interest is now applied to enhancing the network simulation offering of Scala Computing. Ariel co-founded, as Chief Architect, Enfabrica Corporation to drive said efficiencies through new semiconductors. Previously Ariel was an Infrastructure Technologist at Facebook, where he relied on some of the switches and network components created during his prior Distinguished Engineer tenure at Broadcom Corporation.

Duncan Roweth is a Senior Distinguished Technologist at Hewlett Packard Enterprises in the UK. He is working on development of future generations of Slingshot, HPE's Ethernet based network for HPC. Duncan joined HPE with the acquisition of Cray, where he was one of the instigators of the Slingshot program, working on the network components that underpin the US exascale systems and wider HPE/Cray product line.