It's Time for Low Latency

Stephen M. Rumble, Diego Ongaro, Ryan Stutsman,
Mendel Rosenblum, and John K. Ousterhout
Stanford University

Abstract

The operating systems community has ignored network latency for too long. In the past, speed-of-light delays in wide area networks and unoptimized network hardware have made sub-100μs round-trip times impossible. However, in the next few years datacenters will be deployed with low-latency Ethernet. Without the burden of propagation delays in the datacenter campus and network delays in the Ethernet devices, it will be up to us to finish the job and see this benefit through to applications. We argue that OS researchers must lead the charge in rearchitecting systems to push the boundaries of low-latency datacenter communication. 5-10μs remote procedure calls are possible in the short term – two orders of magnitude better than today. In the long term, moving the network interface on to the CPU core will make 1μs times feasible.

1 Introduction

Network latency has been an increasing source of frustration and disappointment over the last thirty years. While nearly every other metric of computer performance has improved drastically, the latency of network communication has not. System designers have consistently chosen to sacrifice latency in favor of other goals such as bandwidth, and software developers have focused their efforts more on tolerating latency than improving it.

Recent developments are finally bringing low latency within reach. Fast communication is available today for those willing to use specialized hardware and software developed by the high-performance computing (HPC) community, and pieces of the low latency puzzle are becoming available for general purpose computing.

In this position paper we argue that it should be possible to achieve end-to-end remote procedure call (RPC) latencies of 5-10μs in large datacenters using commodity hardware and software within a few years. However, achieving this goal will require the creation of a new software architecture for networking with a different division of responsibility between operating system, hardware, and application. In addition, we will probably need new network protocols optimized for low latency in large datacenters. Although several hardware improvements will also be required to reach this goal, the operating system community is in the best position to coordinate all of these changes and create the right end-to-end architecture. Over the longer-term, and with more radical hardware changes (such as moving the NIC onto the CPU chip), we think 1μs datacenter round-trips can be achieved.

	1983	2011	Improved
CPU Speed	1x10Mhz	4x3GHz	$>$ 1,000x
Memory Size	$\leq$ 2MB	8GB	$\geq$ 4,000x
Disk Capacity	$\leq$ 30MB	2TB	$>$ 60,000x
Net Bwidth	3Mbps	10Gbps	$>$ 3,000x
RTT	2.54ms	80μs	32x

Table 1: Network latency has improved far more slowly over the last three decades than other performance metrics for commodity computers. The V Distributed System [5] achieved round-trip RPC times of 2.54ms. Today, a pair of modern Linux servers require 80μs for 16-byte RPCs over TCP with 10Gb Ethernet.

There will be many benefits for datacenter computing if we succeed. Lower latency will simplify application development, increase web application scalability, and enable new kinds of data-intensive applications that are not possible today.

2 A History of High Latency

Network latency has failed to keep up with other improvements in computer performance. Consider technology evolution over the last 30 years (Table 1). In 1983, the V Distributed System [5] had 2.54ms RPC round-trip times on a SUN Workstation [2]. The last three decades have seen only a 30x reduction in latency, while network bandwidth has improved more than 3,000x over the same period, and processor throughput, disk capacity, and memory capacity have also had large gains. Several research projects in the mid- and late-1990s attacked the latency problem [15, 6, 7, 12], but unfortunately their techniques were not adopted by mainstream manufacturers.

Latency is even worse in large datacenters with tens of thousands of servers. Round-trip times are typically 200-500μs [13, 9] and congestion can cause spikes up

Component	Delay	Round-Trip
Network Switch	10-30μs	100-300μs
Network Interface Card	2.5-32μs	10-128μs
OS Network Stack	15μs	60μs
Speed of Light (in Fiber)	5ns/m	0.6-1.2μs

Table 2: Factors that contribute to latency in TCP datacenter communication. "Delay" indicates the cost of a single traversal of the component, and "Round-Trip" indicates the total impact on round-trip time. Messages typically traverse 5 switches in each direction in a large datacenter network and must pass through the OS stack 4 times.

to tens of milliseconds. Table 2 breaks down the major components of latency in datacenters today. Store-and-forward network switches are the largest single contributor, adding tens of microseconds for each hop, but network interface cards and operating system protocol stacks also present major obstacles to low latency. Speed-of-light delays are not a major factor.

High latency is not fundamental or inevitable: designers have chosen to sacrifice latency in order to achieve other goals or work around problems elsewhere in the system. For example, network switches typically employ large packet buffers, which are needed because (a) networks are oversubscribed, resulting in congestion, and (b) the TCP protocol behaves poorly if packets are dropped because of congestion. Unfortunately, the more that buffers are used, the worse latency becomes. Both operating systems and NICs are optimized for bandwidth at the expense of latency; for example, many NICs intentionally delay the delivery of interrupts as much as 30μs in order to allow several packets to be processed with a single interrupt.

3 Impact on Applications

Although it has been convenient for system designers to sacrifice latency, this has not been so convenient for application developers. For example, high latency limits Facebook's applications to 100-150 sequential data accesses within the datacenter for each Web page returned to a browser (any more would result in unacceptable response time for users). As a result, Facebook has resorted to parallel accesses and de-normalized data, both of which add to application complexity [13]. Some features are simply not possible within this constraint.

Because of the complexity of dealing with latency, considerable effort has been expended in recent years to develop application frameworks that are insensitive to latency. For example, MapReduce [10] organizes large-scale applications as a series of parallel stages where data is accessed sequentially in large blocks; as a result, application speed is limited by bandwidth, not latency. However, its dependence on sequential data access makes MapReduce difficult to use for applications that require random accesses. Furthermore, MapReduce is useful only for long-running batch jobs, not for interactive tasks.

High latency rules out entire classes of applications. For example, if an application needs to harness thousands of machines in a datacenter and intensively access random bits of data distributed across the main memories of those machines (e.g., for large-scale graph algorithms), the application is not practical today: it will bottleneck on the network. Network latency was less of an issue in the past when most network requests resulted in disk I/Os, but as the primary locus of data moves from disk to flash or even DRAM, the network is becoming the primary source of latency in remote data accesses.

If latency can be improved by 1-2 orders of magnitude, we believe it will have a revolutionary impact on applications. The most immediate benefit will be for existing applications that suffer from high latency, such as Facebook or Google's statistical machine translation [3]. These applications will become much simpler to develop, since it will no longer be necessary to employ complex workarounds for high latency. They will also become faster and more scalable. More in-depth data exploration will be possible in real-time.

More importantly, we speculate that low latency will enable a new breed of data-intensive applications. The Web has made it possible to assemble enormous datasets, but high latency severely restricts the kinds of operations that can be performed on those datasets, particularly in real time. Low-latency networking will allow more intensive and interactive manipulation of large datasets than has ever been possible in the past. It is hard to predict the nature of these applications, since they could not exist today, but one possible example is collaboration at the level of large crowds (e.g., in massive virtual worlds); another is very large-scale machine learning applications and other large graph algorithms.

The popularity of memcached [1] and "NoSQL" storage systems provides another indication of the benefits of low latency. Their rapid adoption demonstrates just how desirable and powerful fast storage access is. Developers are clamoring for even faster data access, but the network is now the bottleneck: more than 80% of the memcached latency for Facebook is due to the network.

Low latency has the potential to reduce or eliminate other problems that have plagued system designers. For example, many NoSQL systems have sacrificed consistency guarantees by limiting atomic updates to a single row or offering only eventual consistency [11, 4, 8]. Strong consistency is expensive to implement when there are many transactions executing concurrently, since this increases the likelihood of expensive conflicts. In a

system with very low latency, transactions finish more quickly, reducing the overlap between transactions and minimizing conflicts. We speculate that low-latency systems may be able to provide stronger consistency guarantees at much larger system scale.

Low latency may also reduce the incast problems experienced by many applications. With high latency, applications are forced to issue concurrent data requests in order to meet user response deadlines. However, such requests can lead to simultaneous responses, which can cause congestion near the client, overflow small switch buffers, and result in packet loss. This scenario could be avoided if sequential accesses were sufficiently fast that concurrent requests are no longer needed.

4 Low Latency is Within Reach

Several recent developments are putting low latency networking within reach. These include the rise of the datacenter, the next generation of Ethernet switching chips, and faster NICs. The HPC community has shown us that low latency is possible using special-purpose interconnects, but now we are on the brink of achieving the same with commodity hardware.

One of the interesting properties of datacenters is that they pack large amounts of computation and storage close together. While the speed of light limits latency in wide area networks, electrons can traverse 100m of copper cables and back in about 1μs. This means that microsecond-level RPCs are physically possible at very large scale, and the datacenter is the perfect environment to take advantage of low latency networking.

New cut-through switching chips designed for 10Gb Ethernet are dramatically lowering the cost of both latency and bandwidth. For example, switches from Arista, which are based on chips from Fulcrum Microsystems, offer switching delays less than 1μs, which is more than an order of magnitude improvement over the times in Table 2. The cost/port of these switches is still high compared to commodity 1Gb switches, but will drop rapidly over the next few years. Furthermore, the switching chips should get a double benefit from Moore's Law: not only will they improve in latency but the number of ports will increase, which will reduce the number of switching levels required to traverse a datacenter.

The new switching chips also promise to make bandwidth plentiful and cheap. Most existing datacenter networks cannot afford enough bandwidth in the switching fabric for all nodes to communicate randomly at full line rates. As a result, upper layers in the switching fabric are oversubscribed (the total bandwidth of the top layer is typically 100-500x less than the total bandwidth out of individual servers). The resulting congestion can result in delays of tens of milliseconds as packets work their way through deep buffers. The new switching chips are cheap enough to make full bisection bandwidth affordable in datacenter networks, eliminating congestion and the associated delays.

New network interface controllers from companies such as Mellanox are significantly reducing two other major sources of latency. First, they have been optimized to reduce latency within the NIC itself (less than 1μs in each direction). Second, they allow direct access from user space, which eliminates the overhead of passing through the kernel. These interfaces can be used in a polling mode, which also eliminates the latency associated with interrupt handling and context switching. Unfortunately most of these interfaces are designed for specialized interconnects such as Infiniband and Myrinet, so they do not support the standard protocols and APIs expected by most applications.

The HPC community has already produced specialized systems that combine all the benefits above. Using Infiniband switches and NICs from Mellanox, we have measured round-trip times less than 5μs in small-scale networks with reliable delivery protocols analogous to TCP. HPC vendors have demonstrated that low latency is possible, and it seems likely that some of the techniques used in HPC hardware will migrate to mainstream networking. Unfortunately, the interconnects, protocols, and APIs of these systems are very different from the commodity Ethernet/IP/TCP approaches used in most datacenters, so low latency is still beyond the reach of most applications today.

Futhermore, HPC approaches are unlikely to be adopted wholesale. First of all, Ethernet's ubiquity, market dominance, and economies of scale will make it difficult to compete with. Second, the HPC strategy has been to move functions to network interfaces to overcome OS inefficiencies and allow more complex offloading of functionality. This makes the NICs more expensive, but more importantly, it makes the network less flexible. In contrast, Ethernet's simplicity promotes innovation and makes for an excellent research vehicle.

5 The OS Community's Role

Until recently there was little reason for operating systems to worry about latency: external factors such as speed-of-light propagation for long-haul networks and slow switches in datacenters overshadowed any inefficiencies in the operating system. However, the improvements discussed in Section 4 are dramatically reducing the external factors; within a few years the operating system could become the largest remaining obstacle to low latency RPCs in datacenters. Thus, it is now time to rethink the role of the operating system in networking.

As a community, we should set a goal of making

5-10μs RPC times (end-to-end between applications) easily accessible to mainstream datacenter applications within a few years. This section describes some of the issues to address in order to achieve this goal. Section 6 will then argue that we can do even better and should set a longer-term goal of 1μs round-trip times.

The first and most important task is to create a new system architecture for networking with a different division of responsibility between NIC hardware, operating system, and application. The operating system cannot be in the loop for normal message exchanges: data must pass directly between the application and the NIC. We should think of network operations more like memory references and less like disk I/Os: in the same way that the operating system sets up page tables and then lets memory references operate at hardware speeds, the OS should communicate with the NIC to establish mappings for packet demultiplexing (perhaps using mechanisms like those defined for OpenFlow [14]), then get out of the way during normal processing. In addition, the implementation of network protocols may need to be shared between the operating system and applications.

The new networking architecture must also be based on a polling approach to communication, where threads remain on their CPUs while waiting for packets to arrive; it makes no sense to switch contexts during an RPC when the RPC latency is comparable to the context switch time. However, polling may not scale well as more and more applications begin to use it. For example, how should the system behave if there were more threads polling than there were cores/hyperthreads? It may make sense to introduce new synchronization and scheduling mechanisms that combine polling with traditional context switching.

Although future NICs may need to take on some additional functions to enable direct application-level access, in general we argue for *onloading* from the NIC. In recent years some NIC vendors have attempted to offload as much functionality as possible from the CPU to the NIC, including significant portions of network protocols, but we argue that this is the wrong approach. For optimal performance, operations should be carried out on the fastest processor, which is the main CPU; cycles there are now plentiful, thanks to increases in the number of cores. Functionality implemented in the NIC is also harder to change. The NIC should contain the minimum feature set needed to move bits as efficiently as possible between the CPU and the network; all other functions should be implemented in the main processor.

Achieving low latency may also require the development of new network protocols. Our measurements indicate that current TCP implementations account for 25-50μs of latency in round-trip RPC times. Furthermore, TCP is currently optimized for large unidirectional flows rather than small RPC-like exchanges; it is not designed to capitalize on new datacenter switching fabrics (e.g., it behaves poorly if randomized routing is used to minimize congestion); and it does not behave gracefully in the face of incast. We think a two-pronged approach makes sense, where one group of researchers attempts to optimize TCP to minimize its latency and fix its other problems, while a second group makes a clean-slate design of a new network protocol for small low-latency RPC exchanges within large datacenters. One of the advantages of datacenters is that they form their own closed ecosystems: a new protocol can succeed within a datacenter without having to be implemented on every machine in the Internet.

Source of Delay	Quantity, Rate	R-trip Latency
Propagation	50m, 5ns/m	250ns x 2
Transmission	100B, 32Gb/s	25ns x 2
Switching	5 hops, 100ns/hop	500ns x 2
	Total: 1.55μs	

Table 3: End-host processing aside, round-trip network latencies as low as 1.55μs are currently possible for large datacenters. While propagation delays will not improve, we can expect transmission times to drop with higher bitrates, switching latencies to fall, and hop counts to decrease with Moore's Law.

6 Pushing the Envelope: Integrated NICs

5-10μs round-trip latencies seem achievable within a few years, but we believe it is possible to do much better in the longer term. It appears technologically feasible to reduce datacenter RPC latency to 1μs before speed-of-light delays limit further progress. However, 1μs round-trips will require the integration of NIC functionality onto the main CPU die.

Table 3 breaks down network fabric latencies that are achievable today within a 50m diameter using Infiniband QDR switches with 100ns latency and 32Gbps effective line rates. Although propagation delay is significant, it accounts for less than half the total latency. We can expect significant improvements in the 68% of the time spent in transmission and switching delays. For instance, 100Gb Ethernet is on the horizon, and Moore's Law will enable higher switch port densities, which will reduce the total number of hops. In a future scenario with 30ns switch latencies, 8 hops per round-trip, and 100Gbps line rates, round-trip fabric latency could halve to 750ns.

The next major challenge in reducing latency is to eliminate latency on the motherboard. Unfortunately, using off-processor NIC chips introduces significant delays. In order to move data from the CPU to the network, the CPU must flush the data to memory and then the NIC must read the data from memory. Each of these transfers introduces around 100ns of delay, and for most NICs

multiple memory operations are required (e.g., not only must the packet be stored in memory, but separate ring buffer pointers must also be manipulated). Each RPC requires data to pass through NIC chips four times for a total delay of at least 1-2μs. If any direct manipulation of NIC device registers over PCIe is required, such as polling a device register for incoming data, it adds hundreds more nanoseconds of latency.

As a result, 1μs round-trip times cannot be achieved with off-processor NICs. Moreover, there must not be any memory accesses in the fast path: information must move directly between on-chip caches and the network. This will require the integration of NIC functionality onto the main processor chip. Although we realize that such a change will not happen overnight, we argue that fast network communication is as important for large-scale datacenter applications as fast floating-point arithmetic is for scientific applications and that integrating NIC functionality should be a top priority for the processor design community. Using chip real estate for an integrated NIC is likely to improve overall system performance more than adding cores that software developers do not know how to utilize.

We urge everyone in the OS community to apply pressure on hardware architects for integrated NICs, and we believe the OS community should drive the architecture for on-chip networking in order to ensure the best distribution of functionality between hardware, OS, and application. Processor designers are already putting interconnection networks on-die for core-to-core communication – we need to help them think bigger. If we can make the leap to on-chip NICs, 1μs round-trip times could become widely available within ten years.

7 Conclusion

We are on the cusp of a two-order-of-magnitude improvement in the latency of RPC communication. It is time for the operating systems community to implement a new networking architecture and new protocols that solve the latency problem end-to-end and make fast networking easily available to applications. If we can do this, we will not only simplify the development of current applications but also enable new kinds of applications that manipulate large-scale datasets in ways never before imaginable.

8 Acknowledgements

This work was supported in part by the Gigascale Systems Research Center and the Multiscale Systems Center, two of six research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program, and by gifts from SAP, NetApp, Facebook, and Mellanox. This work was also supported in part by an NSERC Post Graduate Scholarship. Nandu Jayakumar provided comments that improved the presentation of the paper.

References

[1] memcached: a distributed memory object caching system, Jan. 2011. `http://www.memcached.org/`.

[2] BECHTOLSHEIM, A. The SUN workstation architecture. Tech. rep., Stanford, CA, USA, 1982.

[3] BRANTS, T., POPAT, A. C., XU, P., OCH, F. J., AND DEAN, J. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (2007), EMNLP-CoNLL '07, pp. 858–867.

[4] CHANG, F., DEAN, J., GHEMAWAT, S., HSIEH, W. C., WALLACH, D. A., BURROWS, M., CHANDRA, T., FIKES, A., AND GRUBER, R. E. Bigtable: a distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7* (Berkeley, CA, USA, 2006), USENIX Association, pp. 15–15.

[5] CHERITON, D. R., AND ZWAENEPOEL, W. The distributed v kernel and its performance for diskless workstations. In *Proceedings of the ninth ACM symposium on Operating systems principles* (New York, NY, USA, 1983), SOSP '83, ACM, pp. 129–140.

[6] CHIOLA, G., AND CIACCIO, G. Gamma: a low-cost network of workstations based on active messages. In *In Proc. Euromicro PDP'97* (1997), IEEE Computer Society.

[7] CHUN, B. N., MAINWARING, A. M., AND CULLER, D. E. Virtual network transport protocols for myrinet. *IEEE Micro 18* (1998), 53–63.

[8] COOPER, B. F., RAMAKRISHNAN, R., SRIVASTAVA, U., SILBERSTEIN, A., BOHANNON, P., JACOBSEN, H.-A., PUZ, N., WEAVER, D., AND YERNENI, R. Pnuts: Yahoo!'s hosted data serving platform. *Proc. VLDB Endow. 1* (August 2008), 1277–1288.

[9] DEAN, J. Keynote talk: Designs, lessons and advice from building large distributed systems. In *The 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware* (October 2009).

[10] DEAN, J., AND GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. *Commun. ACM 51* (January 2008), 107–113.

[11] DECANDIA, G., HASTORUN, D., JAMPANI, M., KAKULAPATI, G., LAKSHMAN, A., PILCHIN, A., SIVASUBRAMANIAN, S., VOSSHALL, P., AND VOGELS, W. Dynamo: amazon's highly available key-value store. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles* (New York, NY, USA, 2007), SOSP '07, ACM, pp. 205–220.

[12] DITTIA, Z. *Integrated hardware/software design of a high performance network interface*. PhD thesis, St. Louis, MO, USA, 2001. AAI3016230.

[13] JOHNSON, R., AND ROTHSCHILD, J. Personal Communications, March 24 and August 20, 2009.

[14] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev. 38* (March 2008), 69–74.

[15] VON EICKEN, T., BASU, A., BUCH, V., AND VOGELS, W. U-net: a user-level network interface for parallel and distributed computing. In *Proceedings of the fifteenth ACM symposium on Operating systems principles* (New York, NY, USA, 1995), SOSP '95, ACM, pp. 40–53.