Efficient Data Transfer Protocols for Big Data

Brian Tierney *, Ezra Kissel†, Martin Swany†, Eric Pouyoul *
* Lawrence Berkeley National Laboratory, Berkeley, CA 94270
† School of Informatics and Computing, Indiana University, Bloomington, IN 47405

***Abstract*—Data set sizes are growing exponentially, so it is important to use data movement protocols that are the most efficient available. Most data movement tools today rely on TCP over sockets, which limits flows to around 20Gbps on today's hardware. RDMA over Converged Ethernet (RoCE) is a promising new technology for high-performance network data movement with minimal CPU impact over circuit-based infrastructures. We compare the performance of TCP, UDP, UDT, and RoCE over high latency 10Gbps and 40Gbps network paths, and show that RoCE-based data transfers can fill a 40Gbps path using much less CPU than other protocols. We also show that the Linux *zero-copy* system calls can improve TCP performance considerably, especially on current Intel "Sandy Bridge"-based PCI Express 3.0 (Gen3) hosts.**

I. Introduction

Modern science is increasingly data-driven and collaborative in nature. Large-scale simulations and instruments produce petabytes of data, which is subsequently analyzed by tens to thousands of scientists distributed around the world. Although it might seem logical and efficient to colocate the analysis resources with the source of the data (instrument or a computational cluster), this is not the typical scenario. Distributed solutions – in which components are scattered geographically – are much more common at this scale, and the largest collaborations are most likely to depend on distributed architectures. Efficient protocols and tools are necessary to move vast amounts of scientific data over high-bandwidth networks in such collaborations.

The Large Hadron Collider[1] (LHC), the most well-known high-energy physics collaboration, was a driving force in the deployment of high bandwidth connections in the research and education world. Early on, the LHC community understood the challenges presented by their extraordinary instrument in terms of data generation, distribution, and analysis.

Many other research disciplines are now facing the same challenges. The cost of genomic sequencing is falling dramatically, for example, and the volume of data produced by sequencers is rising exponentially. In climate science, researchers must analyze observational and simulation data sets located at facilities around the world. Climate data is expected to exceed 100 exabytes by 2020 [9], [13]. New detectors being deployed at X-ray synchrotrons generate data at unprecedented resolution and refresh rates. The current generation of instruments can produce 300 or more megabytes per second and the next generation will produce data volumes many times higher; in some cases, data rates will exceed DRAM bandwidth, and data will be preprocessed in real time with dedicated silicon. Large-scale, data-intensive science projects on the drawing board include the International Thermonuclear Experimental Reactor (ITER)[2] and the Square Kilometre Array[3], a massive radio telescope that will generate as much or more data than the LHC.

In support of the increasing data movement demands, new approaches are needed to overcome the challenges that face existing networking technologies. The ubiquitous Transmission Control Protocol (TCP) is known to have performance problems over long-distance, high-bandwidth networks [8], [19], [23]. With proper tuning, an appropriate congestion control algorithm, and low-loss paths, TCP can perform well over 10Gbps links, but at speeds of 40Gbps and above CPU limitations become an issue. The system overhead of single and parallel stream TCP at 10Gbps is capable of fully using a core on modern processors, raising questions about the viability of TCP as network speeds continue to grow. Also, the administrative burden of ensuring proper TCP tuning settings for various network scenarios has also been a persistent challenge. Alternatives such as user space protocol implementations like UDT [18] provide benefits in certain cases but suffer from increased overhead due to user space buffer copying and context switching, limiting their use for high-performance applications.

Using a combination of intelligent network provisioning and RDMA protocols, one can achieve significant gains over existing methods to support efficient, high-performance data movement over the WAN. Our initial results show that RoCE (RDMA over Converged Ethernet) [5] over layer-2 circuits is worth further evaluation and consideration.

II. Background

A. Science Use Cases

The current state-of-the-art for bulk transfer of large scientific data sets is to use tools such as Globus Online / GridFTP that securely move data using parallel TCP streams. Using these tools on properly tuned hosts, where there are no network impediments such as an under-powered firewall appliance, one should be able to fill a 10Gbps pipe. However, the best per-stream performance one can get from any of todays TCP over socket-based bulk data transfer tools on a 40Gbps host is only around 13Gbps, for reasons described in the evaluation section below. For many of todays use cases this is not an issue, because a) most sites only have a 10Gbps connection, and b) one can do many transfers in parallel to get around this limitation. But in the not too distant future when all big

[1] The Large Hadron Collider: http://lhc.web.cern.ch/lhc/

[2] ITER: http://www.iter.org/

[3] SKA: http://www.skatelescope.org/

data sites have a 100Gbps connections, data sets are 10-100X bigger, and some workflows require the ability to move a single 100TB data set in just a few hours, this per-flow limit will become an issue.

One example of such a workflow is the data processing for the SKA telescope project mentioned above. The current SKA design calls for a total of 2900 sensors in operation and collecting data in 2020, for a total aggregate data rate of 15,000Tbs! The data from the receivers will likely be placed directly in Ethernet frames and sent to a correlator (data processor) for filtering. The resulting filtered data rate will be around 400Tbs, and this data will be sent to a supercomputer center 1000km away for analysis. Clearly this will require hosts with 100Gbps NICs and extremely efficient data transfer protocols.

A more near-term example is the Belle II High Energy Physics experiment currently under construction in Japan[4]. This experiment will generate 1.8GB/s starting in 2016, for a total of 250PB in its first 5 years, and this data will be analyzed by scientists at 65 institutions in 17 countries, with a full copy of the dataset sent to the US. This project will also require more efficient data movement protocols and tools than are available today. Furthermore, the high CPU requirements for standard socket-based TCP mean both higher power consumption and less CPU available for other tasks. As green computing receives more and more attention, the use of more efficient data transfer protocols will be desired.

B. Remote DMA

The InfiniBand Architecture (IBA) [3] and related RDMA protocols have played a significant role in enabling low-latency and high-throughput communications over switched fabric interconnects, traditionally within data center environments. RDMA operates on the principle of transferring data directly from the user-defined memory of one system to another. These transfer operations can occur across a network and can bypass the operating system (OS), eliminating the need to copy data between user and kernel memory space. Direct memory operations are supported by allowing network adapters to register buffers allocated by the application. This "pinning" of memory prevents OS paging of the specified memory regions and allows the network adapter to maintain a consistent virtual to physical mapping of the address space. RDMA can then directly access these explicitly allocated regions without interrupting the host operating system.

Our RDMA implementations make use of the InfiniBand Verbs, *ibverbs*, and RDMA Communication Manager, *rdmacm*, libraries made available within the OpenFabrics Enterprise Distribution [4]. OFED provides a consistent and medium-independent software stack that allows for the development of RDMA applications that can run on a number of different hardware platforms. The IBA specification itself supports both reliable (**RC**) and unreliable (**UC**) RDMA connections. In addition, two different transfer semantics are available: 1) "two-sided" RDMA SEND/RECEIVE references local, registered memory regions which requires posted RECEIVE requests before a corresponding SEND, and 2) "one-sided" RDMA READ/WRITE operations can transfer buffers to and from memory windows whose pointers and lengths have been previously exchanged. The transfer tools developed for our evaluation use RDMA over RC and implement the RDMA WRITE operation due to its lower signaling cost.

The emerging RDMA over Converged Ethernet (RoCE) [5] standard lets users take advantage of these efficient communication patterns, supported by protocols like InfiniBand, over widely-deployed Ethernet networks. In effect, RoCE is InfiniBand protocols made to work over Ethernet infrastructure. The notion of "converged Ethernet", also known as enhanced or data center Ethernet, includes various extensions to the IEEE 802.1 standards to provide prioritization, flow control and congestion notification at the link layer. Since the InfiniBand protocols operate in networks that are virtually loss-free, the motivation for this is clear. The protocol, however, does not directly require any of these extensions and thus it is possible to use RoCE in WAN environments. Until the recent introduction of RoCE, InfiniBand range extenders such as Obsidian Longbow routers were needed to allow for InfiniBand protocols over long distance network paths.

Certain path characteristics are necessary to effectively use the RoCE protocol over wide-area networks. The path should be virtually loss-free and should have deterministic and enforced bandwidth guarantees. Even small amounts of loss or reordering can have a detrimental impact on RoCE performance. Note that the ability to do RoCE also requires RoCE-capable network interface cards (NICs), such as the Mellanox adapters used in our evaluation [1], [28].

C. Transfer Applications

A number of applications were used to collect the results presented below. TCP benchmark results were obtained from both the *netperf* [25] and *nuttcp* [27] tools. RoCE performance results were collected using our own network benchmark called *xfer_test*, which allowed us to compare both TCP and RoCE transfers from the same application. The ability to perform file transfers over RoCE was also built into *xfer_test*.

In addition to the *zero-copy* techniques supported by RDMA protocols, we take advantage of the Linux kernel "splicing" support in our *xfer_test* implementation. The *splice()* and *vmsplice()* system calls use a kernel memory pipe to "splice" or connect file descriptors, which may refer to network sockets, and memory pages while avoiding costly copying to and from user space buffers. The *netperf* tool uses the related *sendfile()* call, which uses *splice*, to provide a *zero-copy* test. The benefit of these approaches is highlighted in our 40Gbps results.

The Globus Toolkit's GridFTP [17] distribution was used to provide performance numbers for a widely-used transfer tool. Our previous work [22] developed an RDMA driver within the Globus XIO [21] framework on which GridFTP is built. However, due to overheads involved in enabling high-performance RDMA within the existing XIO implementation, we focus on our *xfer_test* transfer tool in the following evaluation. Related work [32], [33] has investigated extending

[4] Belle II: http://belle2.kek.jp/

XIO to make it more suitable for RDMA-style communication, and we plan to take advantage of these advances in future testing.

Beyond the logistics involved with traditional host tuning, a number of important considerations govern the ability of RDMA-based applications to achieve expected performance over the WAN, especially as latency increases. It is well understood that the transmission of buffers over the network must be pipelined in order to keep enough data "in-flight" and saturate the capacity of the given network path. TCP solves this using a sliding window protocol tuned to the RTT of the path. In contrast, applications using RDMA are directly responsible for allocating, managing, and exchanging buffers over networks that can have dramatically different requirements, from interconnects with microsecond latencies to RoCE over WANs with 100ms or greater RTTs.

There are two controllable factors that influence the wide-area network performance of an RDMA application: 1) the number and size of buffers, and 2) the number of RDMA operations that are posted, or in transit, at any given time. In the context of an RDMA communication channel, this corresponds to the message size, or size of the memory window in RDMA READ/WRITE operations, and the transmit queue depth, *tx-depth*, respectively. In general, managing fewer, larger buffers can result in less overhead for both the application and the wire protocol, and we developed a configurable ring buffer solution in support of our RDMA implementations that allowed us to experiment with various buffer/message sizes for the transfer of real data. The design of the application itself is tasked with allocating adequate buffers and posting enough RDMA operations based on the characteristics of the network path in order to saturate the given link capacity. Depending on the requirements of a particular application, either RDMA middleware libraries or direct integration of the InfiniBand Verbs API can be used. Although the details of these implementations are outside the scope of this paper, we note that both our tools and commonly available RDMA benchmarks allow for the explicit tuning of message sizes and transmit queue depths.

III. Evaluation

A. *Overview*

We first show our 10Gbps results, then our 40Gbps results.

B. *10Gbps Testing and Analysis*

Our performance analysis of RoCE uses resources from DOE's Advanced Network Initiative[5] (ANI) network and testbed[6], which includes a 100Gbps wave connecting the National Energy Research Scientific Computing Center[7] (NERSC) in Oakland CA to the Argonne Leadership Class Facility[8] near Chicago, IL.

The ANI Testbed, a public testbed open to any researcher, is shown in Figure 1, includes high-speed hosts at both NERSC and ALCF. These results used two hosts at NERSC, each of which has two 6-core Intel processors and 64GB of system memory. These are all PCI Gen-2 hosts, which support a maximum of 27Gbps IO per flow.

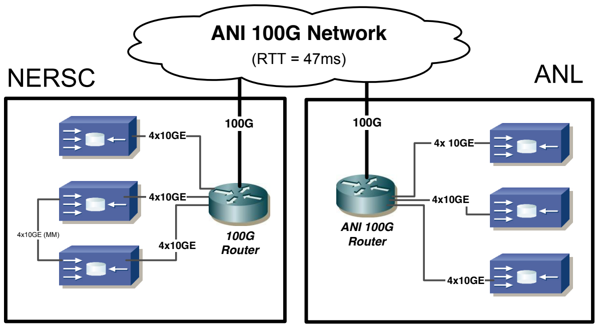


Fig. 1: ANI 100Gbps Testbed Configuration

The NERSC hosts also each have a storage subsystem, consisting of two performance RAID controllers. Each RAID controller has two RAID 0 sets of four drives, for a total of four RAID sets of four drives each. Each RAID set can achieve 400MB/sec sustained read or write, for a total of 1.6GB/sec (12.8Gbps) per host.

All the 10Gbps hosts have a recent 2.6 Linux kernel, and the 40Gbps hosts have a recent 3.3 Linux kernel. We ensured that standard host and NIC driver tuning was performed to ensure the best performance for each of our benchmarks. The Maximum Transmission Unit (MTU) on each installed NIC was set to 9000 bytes and Rx/Tx link layer flow control was enabled by default. For the RoCE tests, the effective MTU is limited to 2048 bytes as defined by the current RoCE specification. Each of our experiments involved memory-to-memory transfers to remove disk I/O as a potential bottleneck.

Note that this testbed, while very high performance, is not a particularly accurate way to emulate a real campus networks. In this testbed the hosts are connected directly to high-end 100Gbps Alcatel-Lucent Model SR 7750 border routers, which have a large amount of buffer space. This may mask some of the issues that would be seen when the endpoints are in a campus network, with less-capable devices in the path, as discussed below.

C. *Transfer Application Baselines*

We ran a series of tests to get baselines for our various applications when running single stream transfer over a 10Gbps NIC with uncapped 100Gbps WAN capacity. These results are summarized in Table I. *nuttcp* was chosen over *iperf* for better reliability when performing UDP benchmarks. For both UDP and TCP, each benchmark was able to achieve 9.9Gbps over the 49ms path. Our RDMA testing with *xfer_test* was able to once again reach 9.7Gbps, which is approximately the maximum achievable "goodput" possible with a 2KB MTU when factoring in protocol overhead.

Single stream GridFTP tests were run for TCP, UDT, and RDMA XIO drivers. Due to a combination of limited memory bandwidth on the ANL Opteron systems and GridFTP/XIO overheads, were were unable to achieve more than 9.2Gbps

[5] Advanced Network Initiative http://www.es.net/RandD/advanced-networking-initiative/

[6] ANI Testbed http://ani-testbed.lbl.gov/

[7] National Energy Research Center http://www.nersc.gov

[8] Argonne Leadership Class Facility http://www.alcf.anl.gov

TABLE I: Application performance baselines for various transfer tools over ANI testbed..

Tool	Protocol	Gbps
nuttcp	TCP	9.9
nuttcp	UDP	9.9
xfer_test	TCP	9.9
xfer_test	RDMA	9.7
GridFTP	TCP	9.2
GridFTP	UDT	3.3
GridFTP	RDMA	8.1

application performance in the TCP case, and much worse performance for both UDT and RDMA. UDT performance is particularly poor due to the way it interacts with the XIO framework, requiring extra memory copying. Due to these factors, our remaining 10Gbps evaluation reports results for the *xfer_test* benchmark and uses *nuttcp* to introduce competing TCP and UDP traffic.

D. RoCE with Competing Flows

An open question has been how RoCE behaves when sharing paths with competing traffic. Our ability to perform path provisioning over a configurable testbed environment gave us an opportunity to understand how well RoCE performs when sharing capacity with commodity TCP/UDP flows as well as other competing RoCE transfers. Given the lack of converged Ethernet support in the ANI testbed, we hypothesized that competing traffic over shared bottlenecks would significantly impact RoCE performance, if not completely impair the ability to perform a successful RoCE transfer.

Our first test looked at this latter scenario where we investigate two 10Gbps RoCE transfers (two pairs of independent 10Gbps RoCE-capable NICs on separate hosts) interacting over both an unconstrained and bandwidth capped circuit. Figure 2 shows that RoCE equally shares the available capacity in both cases. With a 20Gbps WAN path, each RoCE flow easily reaches 9.7Gbps over our 120 second test. When the WAN path is capped to 10Gbps, we see each RoCE flow evenly split the available capacity for the duration of the transfer, and the flows do not interfere with each other. Surprisingly, these results ran counter to our original hypothesis.

We expect this fair sharing between flows to continue as more RoCE transfers are added as long as layer-2 flow control (i.e. 801.2d PAUSE frames) remains enabled on both end hosts and intermediate network elements. In the absence of link layer flow control, OSCARS would need to provision dedicated paths (e.g. separate VLANs) for each RoCE flow, or some other QoS mechanism would be required to classify lossless traffic from other flows, e.g, matching on the RoCE Ethertype (0x8915).

Our next set of tests explored RoCE behavior when we introduced a number of TCP flows competing for the available shared link capacity. In this scenario, we had a 20Gbps cap on the WAN path and the two 10Gbps RoCE flows as described above, and we included an additional host with two 10Gbps NICs that could generate a total of 20Gbps TCP traffic over our shared path. This resulted in a total of 40Gbps (20Gbps RoCE and 20Gbps TCP) competing for 20Gbps over the WAN. Three commonly deployed TCP congestion control algorithms were investigated: Reno, CUBIC, and HTCP. In each experiment, we varied the number of parallel streams from 1 to 32, which were distributed evenly across the two TCP traffic-generating NICs. In addition, we explored how limiting the TCP congestion window to 16MB affected the RoCE flows.

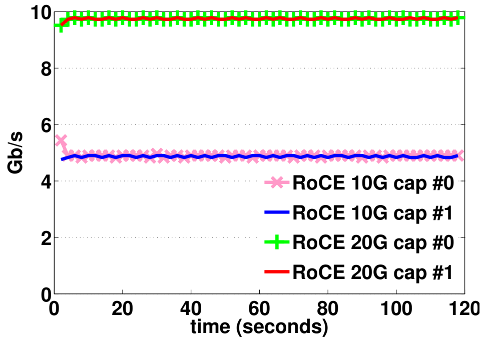


Fig. 2: Two 10Gbps RoCE transfers with 10Gbps and 20Gbps bandwidth caps.

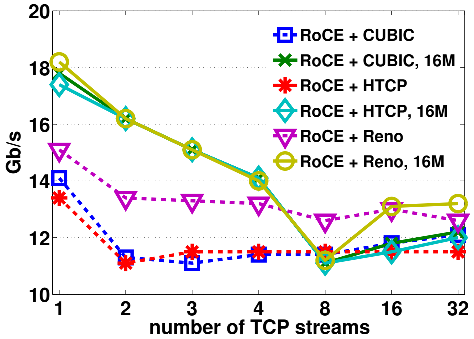


Fig. 3: RoCE performance with increasing number of TCP streams.

Figure 3 shows combined RoCE performance when competing with each of the TCP cases just described. The overall results show that RoCE maintains well over 50% of the available link capacity in all cases; however, there are some subtle variations. When TCP is not limited by a small congestion window, it has a relatively consistent influence on the RoCE flows. Both CUBIC and HTCP have similar effects while Reno is shown to be less aggressive. The RoCE performance for the single stream TCP case is significantly higher since we have 10Gbps TCP instead of 20Gbps TCP in this one case. In all other cases, both 10Gbps TCP NICs were involved. As more streams are added in the CUBIC and HTCP cases, we see that the parallel streams begin to interfere with one another, giving a slight boost to RoCE performance. The opposite is true for Reno where it is able to achieve slightly higher aggregate rates with additional streams, although still not as aggressive as CUBIC or HTCP.

As expected, a small TCP congestion window limited the performance for fewer numbers of TCP streams. At 8 parallel streams and 16MB windows, we see the aggregate rate matching that of the unconstrained, or autotuned, streams

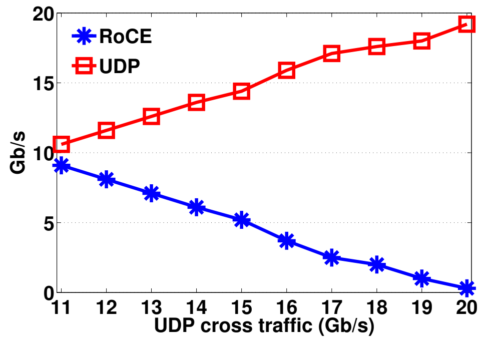


Fig. 4: RoCE performance with competing UDP traffic.

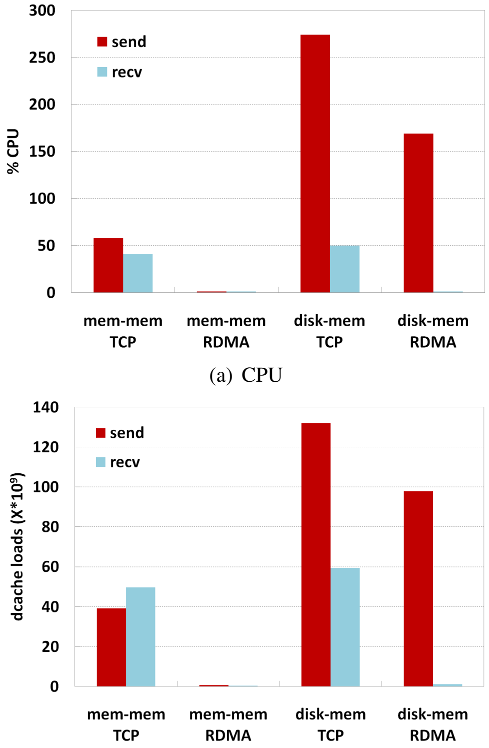
where Reno now matches the performance of both CUBIC and HTCP for this particular 49ms path. As the number of streams increase, we see the 16MB window cases continue to follow the trend of the autotuned streams as each was able to “ramp up” and remain in congestion avoidance with the side effect of influencing other flows.

The situation changes dramatically when we introduced increasing amounts of rate-controlled UDP traffic. As shown in Figure 4, UDP effectively takes as much bandwidth as requested, limiting the ability for the two 10Gbps RoCE flows to remain competitive. In a few experiments when blasting 20Gbps UDP, we observed that one or both of the RoCE transfers would stall, requiring a restart of the transfer tool. We speculate that this is due to overflowing queues in the transit network equipment, causing loss of RoCE frames before the link layer flow control can reduce the sending rate of the RoCE NICs.

E. Overhead Analysis

Beyond the potential to provide consistently good throughput over WANs, RoCE gives high-performance transfer applications a mechanism to move large data sets with very little system overhead. This fact ideally enables an efficient overlap between communication and computation for a number of data-intensive use cases. In our SC11 testbed, we have shown how our *xfer_test* tool can transfer at 9.7Gbps with as little as 1% CPU overhead. In this section, we extend our analysis to look at both CPU and memory bus overheads for memory-to-memory and disk-to-memory transfers.

The memory-to-memory tests were run as a 10Gbps RoCE transfer using *xfer_test* between a sending and receiving host, exactly the same as in our performance evaluation described above. For the disk-to-memory transfers, we took advantage of the disk subsystem available on the ANI testbed hosts. The disk array configuration provided in excess of 10Gbps read throughput over 4 separate partitions, requiring the use of 4 instances of *xfer_test* running in parallel to perform an equivalent 10G transfer test. We aggregated the results obtained from each running instance in this case.



(a) CPU

(b) Data Cache Loads

Fig. 5: CPU and data cache overhead for TCP and RDMA transfers, both memory-to-memory and disk-to-memory.

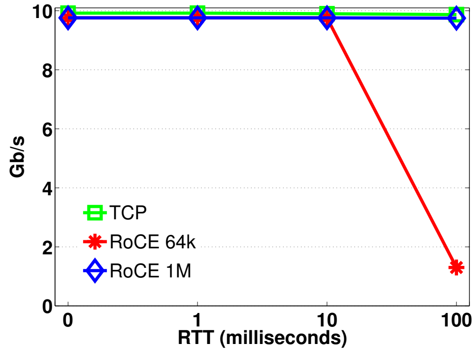
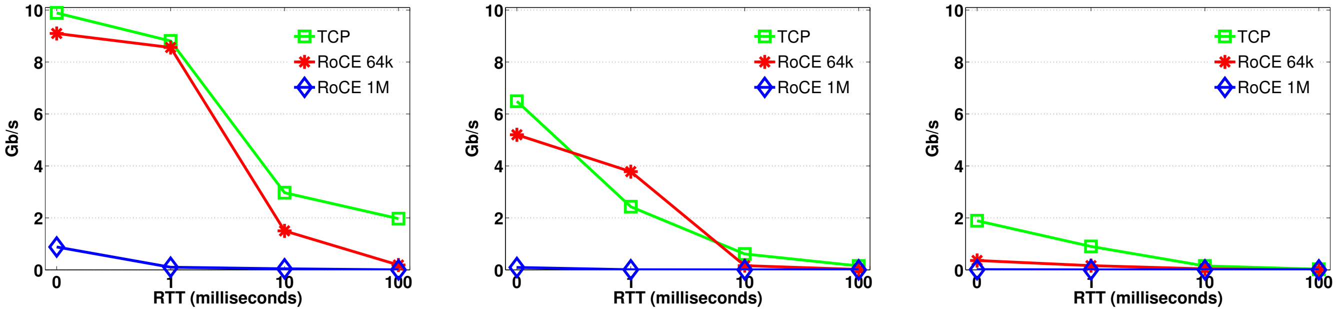


Fig. 6: RoCE and TCP performance over a clean path.

Our CPU numbers were collected using the *nmon* [26] system monitoring utility. We report CPU load percentage as a cumulative value over the number of cores present on the system. For example, an 8 core system could have up to 800% total CPU load. To get a sense of memory bus usage, we took advantage of the Linux *perf* tool [15] to profile our application and report data cache load instructions issued by the CPU over the duration of the transfer.

Figure 5 shows both the CPU load and number of data cache loads used in each case, for both sending and receiving side. For memory-to-memory transfers, the RoCE transfer provides up to a 50X reduction in CPU usage and data cache instructions compared to the TCP case within the same application, again using around 1% of a single core. The disk-to-memory results were dominated by the I/O operations involved in reading from the disk subsystem. However, we note that RDMA effectively eliminates the network communication overhead in this case, significantly reducing both the

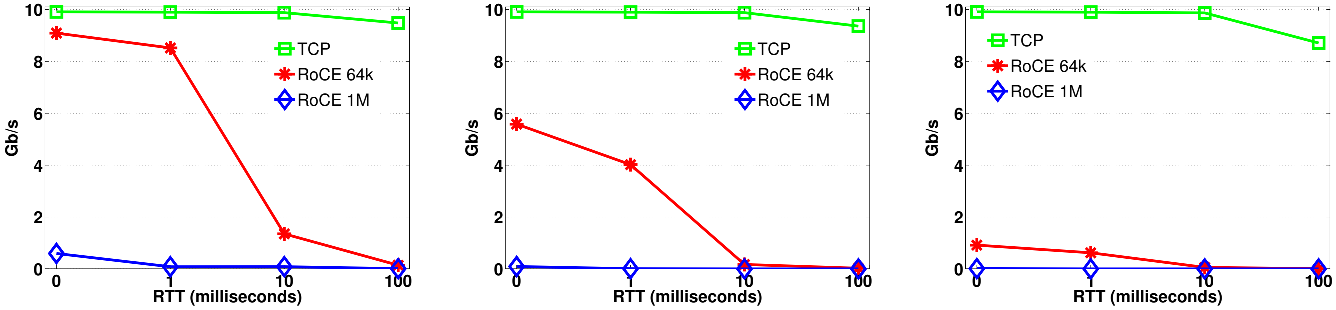


(a) .001% loss

(b) .01% loss

(c) .1% loss

Fig. 7: Comparing RoCE performance with TCP over increasing loss.



(a) .001% reordering

(b) .01% reordering

(c) .1% reordering

Fig. 8: Comparing RoCE performance with TCP over increasing reordering.

CPU and memory usage. We also note that our receive side CPU values are slightly higher in the disk-to-memory tests even though we are similarly writing to memory as in the memory-to-memory tests. This is due to the fact that we have 4 instances of *xfer_test* running in parallel in the disk-to-memory tests, which naturally increases system overhead and inflates our aggregate numbers. In future testing, we hope to evaluate a disk subsystem capable of sustaining 10Gbps+ sequential reads and writes in addition to an extended version of our benchmark that operate across parallel data sources.

F. RoCE over Impaired Conditions

Poor network conditions can have a large impact on RoCE performance. This fact has motivated the need for the provisioning capabilities described above. To better quantify the impact of less than ideal network conditions on RoCE transfers, we made use of a Spirent XGEM [2] network impairment device. Installed between two end hosts in our lab, the XGEM allowed us to evaluate the performance of both RoCE and TCP transfers while emulating a wide variety of WAN latency, loss, and reordering cases. Distinct from our previous two testbed environments, the systems used for these experiments were compute nodes with 8-core Intel Xeon CPUs and 189GB of RAM. Each ran a recent Linux 2.6 kernel with CUBIC as the default congestion control algorithm, and appropriate host and NIC driver tuning was also applied.

In Figure 6, we show that near line-rate RoCE and TCP performance can be achieved over clean, loss-free conditions. This graph also demonstrates the impact that message size and transmit queue depth can have on RDMA transfer performance over high-latency paths. In each of the experiments that involved the XGEM, our RoCE benchmark used a message size of either 64KB or 1MB with a *tx-depth* of 250. Smaller messages have the benefit of incurring less costly retransmissions due to loss or reordering are less costly. On the other hand, there is also increased overhead from the application's perspective in managing many small buffers. This also requires a much larger transmit queue depth for high-latency paths in order to keep the network saturated. This tradeoff is made apparent in the figure where we see less than 2Gbps performance in the 64KB case at 100ms RTT.

Figures 7 and 8 show the impact of both loss and reordering on RoCE and TCP performance, respectively. Increasing amounts of loss severely impact both TCP and RoCE performance as path latency increases. Reordering has less of an impact on TCP where reassembly of the byte stream is possible in the receive buffer; however, RoCE is affected almost identically as in the loss cases. We also see that RoCE transfers with larger RDMA message sizes are much more prone to performance issues over even slight impairments and little to no latency.

G. 40Gbps Testing and Analysis

Our 40Gbps testing used two Intel Sandy Bridge[9] hosts with PCIe 3.0[10] and Mellanox 40Gbps NICs. Each NIC is directly connected to the 100Gbps router, and a 40Gbps layer 2 circuit

[9] Sandy Bridge Architecture: http://en.wikipedia.org/wiki/Sandy_Bridge/

[10] PCIe 3.0: http://en.wikipedia.org/wiki/PCI_Express#PCI_Express_3.0

TABLE II: 40Gbps performance results.

Tool	Protocol	Gbps	Send_CPU	Recv_CPU
netperf	TCP	17.9	100%	87%
netperf	TCP-sendfile	39.5	34%	94%
netperf	UDP	34.7	100%	95%
xfer_test	TCP	22	100%	91%
xfer_test	TCP-splice	39.5	43%	91%
xfer_test	RDMA	39.2	2%	1%
gridftp	TCP	13.3	100%	94%
gridftp	UDT	3.6	100%	100%
gridftp	RDMA	13	100%	150%

was configured to go from NERSC to ALCF and back, for a total round trip latency of 94 ms.

Our methodology for 40Gbps was the same as with 10Gbps above. One key addition is the use of *zero-copy* TCP tests to highlight the limitations of traditional TCP transfers at 40Gbps speeds. We also make use of *netperf* in place of *nutttcp* due to improved *sendfile()* and UDP benchmarking support. A summary of each of our bandwidth tests is shown in Table II including both the sender and receiver CPU utilization as reported by *nmon*. Given the Non-Uniform Memory Architecture (NUMA) of our 40Gbps test hosts, we also ensured that each application was bound to the appropriate NUMA node to prevent CPU migrations, in in most cases binding the thread of execution to a single core on both the sender and receiver. For tests that use multiple threads (GridFTP, disk-to-mem), the applications were able to bind to multiple cores within a particular NUMA node.

In each of our three TCP application tests, none were able to achieve more than 45% of the available link bandwidth. The best result was obtained by *xfer_test* at 22Gbps for a single stream. The limiting factor is the overhead involved in copying between kernel and user space buffers, and indeed our *perf* profiles showed a majority of CPU time spent performing memory copies. A clear advantage of using the *zero-copy* calls in our benchmarks is the reduction in this memory copy overhead, allowing single stream TCP performance to saturate the 40Gbps links with significantly reduced CPU utilization at the sending side (TCP-sendfile and TCP-splice cases).

GridFTP is again limited by the overheads within the current XIO framework and achieves 13Gbps for both TCP and RDMA cases. As opposed to sending a fixed amount of allocated memory as in our benchmarks, GridFTP performs additional copying into I/O vectors that get passed along the XIO driver stack. In the RDMA case, the requirement to copy these vectors into ring buffers within the XIO driver also limits the performance we can achieve. We anticipate that solutions such as EXIO as mentioned above, and the addition of *zero-copy* support within XIO, will have a considerable impact on single-stream GridFTP performance over high-speed networks. Finally, we see that GridFTP using UDT achieves a modest gain from the faster 40Gbps hosts but remains an unviable option with the current implementation. Note that single stream GridFTP was use for all results in this paper. Slightly higher throughput was observed with multiple streams, but the goal of this paper was to directly compare single stream TCP with RoCE.

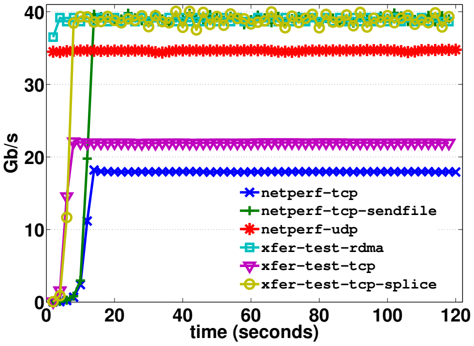


Fig. 9: Performance of 40Gbps benchmarks over 120 seconds.

The *xfer_test* RDMA test is the clear winner in terms of achieving single-stream 40Gbps performance with minimal system overhead. As in our 10G testing, reaching 40Gbps speeds using RoCE involves ensuring the application has allocated and posted adequate buffers to match the network characteristics. At 94ms, the BDP of the path approaches 500MB, which requires the application to stage a substantial amount of data to sustain 40Gbps throughput. We note that our 40Gbps profiling results mirrored those performed at 10Gbps with relatively similar reductions in CPU and memory utilization across tests.

For each of our benchmark tests, *netperf* and *xfer_test*, we plot the application "goodput" over 120s from start to finish in Figure 9. With CUBIC and proper host tuning applied, each TCP test is able to ramp-up in a relatively short amount of time, under 20 seconds for each test. The RDMA transfer is able to saturate the 40Gbps link nearly immediately and provide the benefit of full link utilization while minimizing wasted time on the network.

Lastly, we revisited the file transfer scenario from our earlier 10Gbps tests. Given the lack of a disk subsystem capable of sustained throughput at 40Gbps, we relied on a memory-backed filesystem, or "ramdisk", to evaluate how our *xfer_test* tool could transfer a large data set. When reading from a file descriptor, our tool invokes an additional thread of execution that consumes extra CPU cycles and adds to memory bus contention as the file contents is copied into a ring buffer before being sent over the network. For the disk-to-mem tests using TCP, the extra read overheads limited the transfer rate to 16Gbps. The reduction in overheads for each of the TCP-splice and RDMA cases improved this rate to 33Gbps. This is effectively how quickly our tool was able to read from the ramdisk into the ring buffer while simultaneously writing to the network. Additional improvements could be made in intelligently staging file data from disk into memory, and in particular when adapting to the specifics of an actual disk subsystem.

H. Summary of Results

Our evaluation, covering a number of experiments, has shown that RoCE provides consistently good performance while maintaining low system overhead for our transfer application. Our experiments relied on a static network configuration with the ability to enforce bandwidth caps over the WAN path, which allowed us to evaluate RoCE with competing traffic. Our lab testbed used an XGEM network impairment device to demonstrate how relatively subtle adverse network conditions can negatively affect RoCE performance, motivating the need for building dedicated paths.

Contrary to our initial expectations, RoCE was able to perform well with competing traffic over both shared and bandwidth constrained paths. With standard layer-2 flow control enabled, we were able to simultaneously send both TCP and other RoCE flows without seriously impairing RoCE performance. Indeed, the RoCE transfers were able to compete favorably with multiple parallel TCP streams.

However, we note that in many typical network deployments, link layer flow control may not be possible to enable on all segments, especially at the edge of the network. When testing with flow control disabled across the end host NICs in the ANI testbed, we experienced much less robust behavior in our RoCE transfers. In many cases, our RoCE tests would stall when any significant amount of competing traffic was introduced. We intend to extend our evaluation to explore the role of both existing 802.1d flow control and converged Ethernet extensions on RoCE transfers. In the absence of any form of link layer flow control, our conclusion is that layer-2 services like OSCARS are necessary in order to differentiate lossless traffic from other best-effort flows, particularly if the best possible RoCE performance is desired.

Finally, we have shown that RoCE can scale well to 40Gbps speeds over WAN paths. While other solutions that provide good levels of TCP performance do exist, the ability of RoCE to saturate high-latency WAN paths in conjunction with very low system overhead makes it a compelling technology for use in future network applications.

IV. Related Work

Being a recently proposed standard, there has been relatively little previous research in analyzing RoCE performance over existing Ethernet infrastructure. A number of other RDMA and *zero-copy* protocols not involving InfiniBand (IB) have been proposed to run over Ethernet. These include technologies such as Intel's Direct Ethernet Transport (DET) [6] and approaches that use iWARP-enabled NICs [14], [28]. Compared to RoCE and Infiniband, DET does not provide full OS-bypass functionality with limited hardware support, while iWARP remains bound to the limitations of TCP/IP.

On the other hand, there have been active efforts involved with extending IB fabrics over WANs [12], [24] and comparisons of IB to existing 10Gbps Ethernet in high-latency transfer scenarios [29]. These evaluations rely on IB extension devices which limit WAN performance to approximately 8Gbs, whereas our approach shows that RoCE can easily saturate existing 10Gbps networks. Other related work has investigated RDMA-capable storage protocols over WANs [10], [31] and explored system-level benefits of RDMA interfaces over 10Gbps networks [7].

Considerable efforts have been made in modifying TCP's AIMD-based congestion control algorithm, resulting in numerous variations for improving performance over WANs. These include High-Speed TCP (HSTCP) [16] and FAST TCP [30], among many others too numerous to mention here. Others have investigated user space implementations such as UDT [18], which provides reliability over UDP but suffer from increased overhead, limiting their practical deployment in high-speed networking applications. Newer transport protocols such as SCTP [11] seek to improve performance through multistreaming, while others allow features such as explicit congestion feedback [20]. In all of these cases, improvements have been incremental while failing to address new modes of thinking in the transport of bulk data over long distances.

V. Conclusion

Scientific data sets are growing at exponential rates, and new data movement protocols and system interfaces are needed to keep up. TCP and UDP using traditional UNIX sockets use too much CPU to be able to scale to the data rates needed for tomorrows scientific workflows.

Our experience on ESnet's 100Gbps testbed have shown that alternative data movement solutions exist that work much better than TCP/UDP over sockets. The first improvement is possible through use of *zero-copy* system calls instead of traditional *send()/recv()*, which can double your performance using the latest versions of Linux on the latest hardware. The second alternative is to use RoCE, which uses very little CPU and should be able to scale well beyond speeds of 40Gbps. However RoCE requires hardware support in the NIC, and a congestion free layer-2 circuit to work well, but this may be fairly common in the future.

Our experiments evaluated RoCE flows in a number of scenarios, including multiple RoCE flows in a single circuit, and a mix of RoCE, TCP, and UDP flows in a single circuit. We found that multiple RoCE flows work well over a single shared path. This means that cluster-to-cluster transfers using RoCE should be able to provision a single circuit, and not require multiple circuits with additional configuration complexity.

We were surprised to learn that, to a point, RoCE and TCP can co-exist on the same circuit rather well, at least in our test environment. We believe this was due to a combination of layer-2 flow control and the deep buffers available in the 100Gbps routers we were using, thus ensuring that there was very little loss.

VI. Acknowledgments

We would also like to thank Bob Pearson and David McMillen, who helped us with RoCE tuning issues, and Chin Guok of ESnet for helping with router configuration. This work was supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 and DE-SC0001421. This research used resources of the ESnet Advanced Network Initiative (ANI) Testbed, which is supported by the Office of Science of the U.S. Department of Energy under the former contract number above, funded through the The American Recovery and Reinvestment Act of 2009.

References

[1] Connectx-2 EN with RDMA over Ethernet (RoCE). http://www.mellanox.com/related-docs/prod_software/ConnectX-2_RDMA_RoCE.pdf.

[2] Gem: Network impairment emulator. http://www.spirent.com/Solutions-Directory/Impairments_GEM.

[3] Infiniband trade association. http://www.infinibandta.org.

[4] Ofed: Open fabrics enterprise distribution. https://openfabrics.org/ofed-for-linux-ofed-for-windows/ofed-overview.html.

[5] Infiniband. architecture specification release 1.2.1 annex a16: Roce. Infiniband Trade Association, 2010.

[6] Intel direct ethernet transport (det). http://software.intel.com/en-us/articles/intel-direct-ethernet-transport, 2010.

[7] P. Balaji. Sockets vs rdma interface over 10-gigabit networks: An in-depth analysis of the memory traffic bottleneck. In *In RAIT workshop .04*, page 2004, 2004.

[8] C. Barakat, E. Altman, and W. Dabbous. On tcp performance in a heterogeneous network: A survey. *IEEE Communications Magazine*, 38:40–46, 2000.

[9] BES Science Network Requirements, Report of the Basic Energy Sciences Network Requirements Workshop. Basic Energy Sciences Program Office, DOE Office of Science and the Energy Sciences Network, 2010.

[10] B. Callaghan, T. Lingutla-Raj, A. Chiu, P. Staubach, and O. Asad. Nfs over rdma. In *Proceedings of the ACM SIGCOMM workshop on Network-I/O convergence: experience, lessons, implications*, NICELI '03, pages 196–208, New York, NY, USA, 2003. ACM.

[11] A. L. Caro, Jr., J. R. Iyengar, P. D. Amer, S. Ladha, G. J. Heinz, II, and K. C. Shah. Sctp: A proposed standard for robust internet data transport. *Computer*, 36:56–63, November 2003.

[12] S. Carter, M. Minich, and N. S. V. Rao. Experimental evaluation of infiniband transport over local- and wide-area networks. In *Proceedings of the 2007 spring simulation multiconference - Volume 2*, SpringSim '07, pages 419–426, San Diego, CA, USA, 2007. Society for Computer Simulation International.

[13] D. N. Williams et al. Data Management and Analysis for the Earth System Grid. Journal of Physics: Conference Series, SciDAC 08 conference proceedings, volume 125 012072, 2008.

[14] D. Dalessandro and P. Wyckoff. A performance analysis of the ammasso rdma enabled ethernet adapter and its iwarp api. In *In Proceedings of the IEEE Cluster 2005 Conference, RAIT Workshop*, 2005.

[15] A. C. de Melo. The new linux 'perf' tools. http://vger.kernel.org/~acme/perf/lk2010-perf-paper.pdf, 2010.

[16] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649 (Experimental), December 2003.

[17] GridFTP. http://www.globus.org/datagrid/gridftp.html.

[18] Y. Gu and R. Grossman. Udt: Udp-based data transfer for high-speed wide area networks. *Computer Networks (Elsevier) Volume 51, Issue 7*, 2007.

[19] V. Jacobsen, R. Braden, and D. Borman. Tcp extensions for high performance. RFC 1323, May 1992.

[20] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '02, pages 89–102, New York, NY, USA, 2002. ACM.

[21] R. Kettimuthu, W. Liu, J. M. Link, and J. Bresnahan. A gridftp transport driver for globus xio. In *PDPTA*, pages 843–849, 2008.

[22] E. Kissel and M. Swany. Evaluating high performance data transfer with rdma-based protocols in wide-area networks. In *Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications (HPCC-12)*, 2012. To appear.

[23] S. Molnár, B. Sonkoly, and T. A. Trinh. A comprehensive tcp fairness analysis in high speed networks. *Comput. Commun.*, 32(13-14):1460–1484, 2009.

[24] S. Narravula, H. Subramoni, P. Lai, R. Noronha, and D. K. Panda. Performance of hpc middleware over infiniband wan. In *Proceedings of the 2008 37th International Conference on Parallel Processing*, ICPP '08, pages 304–311, Washington, DC, USA, 2008. IEEE Computer Society.

[25] netperf. http:/www.netperf.org/.

[26] nmon. http://nmon.sourceforge.net/.

[27] nuttcp. http://wcisd.hpc.mil/nuttcp/Nuttcp-HOWTO.html.

[28] M. Oberg, H. M. Tufo, T. Voran, and M. Woitaszek. Evaluation of rdma over ethernet technology for building cost effective linux clusters. In *7th LCI International Conference on Linux Clusters: The HPC Revolution*, 2006.

[29] N. S. V. Rao, W. Yu, W. R. Wing, S. W. Poole, and J. S. Vetter. Wide-area performance profiling of 10gige and infiniband technologies. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 14:1–14:12, Piscataway, NJ, USA, 2008. IEEE Press.

[30] D. X. Wei, C. Jin, S. H. Low, and S. Hegde. Fast tcp: motivation, architecture, algorithms, performance. *IEEE/ACM Trans. Netw.*, 14(6):1246–1259, 2006.

[31] W. Yu, N. Rao, P. Wyckoff, and J. Vetter. Performance of rdma-capable storage protocols on wide-area network. In *3rd Petascale Data Storage Workshop PSDW 2008*, 2008.

[32] W. Yu, Y. Tian, J. Vetter, N. S. Rao, T. Liu, and G. Shainer. Exio: Enabling globus on rdma networks - a case study with infiniband. Technical Report AU-CSSE-PASL/10-TR01, PASL, Dept. of CDDSSE, Auburn University, 2010.

[33] W. Yu, Y. Tian, and J. S. Vetter. Efficient zero-copy noncontiguous i/o for globus on infiniband. In *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops*, ICPPW '10, pages 362–368, Washington, DC, USA, 2010. IEEE Computer Society.