Parallel Computing 48 (2015) 15–39

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Scalable connectionless RDMA over unreliable datagrams

Ryan E. Grant[a,*], Mohammad J. Rashti[a,1], Pavan Balaji[b], Ahmad Afsahi[a]

[a] *Department of Electrical and Computer Engineering, 19 Union Street, Walter Light Hall, Queen's University, Kingston, ON K7L 3N6, Canada*
[b] *Mathematics and Computer Science Division, Argonne National Laboratory, Bldg. 240, Rm. 3146, Argonne, IL 60439, USA*

A R T I C L E I N F O

Article history:
Received 29 May 2013
Revised 19 February 2015
Accepted 20 March 2015
Available online 11 April 2015

Keywords:
iWARP
RDMA
Datagrams
Unreliable network transport
Ethernet
Datacenter

A B S T R A C T

The overhead imposed by connection-based protocols for high-performance computing (HPC) systems can be detrimental to system resource usage and performance. This paper demonstrates for the first time a unified send/recv and Remote Direct Memory Access (RDMA) Write over datagrams design for RDMA-capable network adapters. We previously designed the first and only unreliable datagram RDMA model, RDMA Write-Record, and demonstrated its superior performance over connection-based RDMA. RDMA Write-Record can be applied to several RDMA capable networks, such as iWARP and InfiniBand (which does not support unreliable RDMA Writes). iWARP is a state-of-the-art, high-speed, connection-based RDMA networking technology for both local and wide-area Ethernet networks. iWARP is used as the platform to demonstrate our unreliable RDMA operation design for both channel and memory semantics. We previously outlined the requirements for extending iWARP to operate over datagrams. Here we extend our work on commercial datacenter applications by providing broadcast support for send/recv. In order to study the scalability of datagram-iWARP, we added Message Passing Interface support for RDMA Write-Record to investigate the scalability of HPC-based scientific applications for both send/recv and RDMA Write-Record. The results show that both models outperform their connection-based alternatives, providing superior performance and scalability in a software prototype.

1. Introduction

Unreliable network transports are an important mechanism for providing high-performance and low-latency communications. Unreliable transports (such as UDP) operate over many networks including Ethernet networks, which are some of the most ubiquitous computer networks in use today. Ethernet's wide adoption has made such networks available at lower cost than that of competing high-speed networks, despite lagging behind in cutting-edge network performance.

Other high-speed networks such as InfiniBand also provide unreliable transports. However, none of the unreliable network transports available today support one-sided Remote Direct Memory Access (RDMA) operations. InfiniBand provides support for two-sided send/recv operations but does not specify one-sided *put* and *get* operations over unreliable datagrams. iWARP, which defines RDMA functionality of Ethernet networks, provides operations only over reliable transport such as TCP and SCTP. Other network specifications such as Portals (used in Cray interconnects) do not currently support unreliable transports.

[*] Corresponding author at: Sandia National Laboratories, Scalable Systems Software Department, P.O. Box 5800 Albuquerque, NM 87185-1319, United States. Tel.: +505 844 1252; fax: +505 284 2518.

E-mail addresses: regrant@sandia.gov (R.E. Grant), mohammad.rashti@queensu.ca, mrashti@rnet-tech.com (M.J. Rashti), balaji@mcs.anl.gov (P. Balaji), ahmad.afsahi@queensu.ca (A. Afsahi).

[1] Present Address: RNET-Technologies, Inc. 240 W, Elmwood Dr. Suite 2010, Dayton, OH 45459-4248.

http://dx.doi.org/10.1016/j.parco.2015.03.009
0167-8191/© 2015 Elsevier B.V. All rights reserved.

Because of the large existing Ethernet infrastructure, it is desirable to upgrade existing networks piece by piece rather than overhauling networks by complete networking infrastructure replacement. Therefore, a high-speed Ethernet networking solution that can leverage the performance benefits of advanced networks such as operating system bypass, RDMA, and offloading for both TCP and UDP traffic is attractive.

Some existing solutions to high-speed Ethernet have taken advantage of offloading technologies, with many offering abilities such as stateless offloading, performing segmentation on the network adapter and calculating checksums. Solutions for stateful protocol offloading also exist, typically referred to as TCP offload engines, or TOEs. Existing iWARP-compatible network adapters offer stateless and stateful offloading capabilities in addition to RDMA capabilities, enabling them to perform zero-copy-based communications as well as bypassing the operating system, providing, greater CPU availability, increased network throughput, and decreased latency [15]. However, existing iWARP networking hardware provides these advanced features only for TCP-based traffic.

Current high-performance networks do not allow for RDMA Write/Read operations to occur over a connectionless and/or unreliable transport. Moreover, they typically require that the underlying networking layers provide a reliable, in-order delivery of data (with the exception of InfiniBand's unreliable datagrams, which allow unreliable send/recvs). This requirement has several limitations compared with a design that allows for unreliable data transmission and does not require the storage and manipulation of data associated with ongoing network connections. For very large systems or for systems serving a very large number of clients, the administration and storage of the data associated with these many individual connections can be onerous. Existing high-speed networking technologies such as InfiniBand [24] have acknowledged such limitations in connection-based networks by introducing technologies such as eXtended Reliable Connections (XRC) [24]. These technologies help mitigate the overhead incurred as a result of connections. Connectionless approaches eliminate this concern to a greater extent, by not incurring the overhead of connections in the first place.

A datagram-based solution offers flexibility to application developers in adapting the overhead of the network directly to their individual needs. For example, for a streaming application that requires time-dependent data, a reliable data stream may not be required because any data that had to be retransmitted would no longer be current enough to be of use. Particularly, communication jitter and the sustained throughput are more important for such applications, than the data transfer reliability and latency. Alternatively, an application could require reliable communication but not need the flow control capabilities of TCP [18]. The reduced complexity of the datagram approach provides for faster communication and lower overhead than either of the current iWARP alternatives of TCP or Stream Control Transmission Protocol (SCTP) [34]. A TCP-based implementation can suffer high jitter and low throughput in a long-haul network with potential errors (such as loss, jitter and reordering).

Datagram-based iWARP also reduces the complexity of the networking stack by not requiring use of the Marker PDU Alignment layer (MPA) [5], since middle-box fragmentation is not an issue when going over datagram transports. This feature is helpful in enhancing performance (both latency and capacity) as well as reducing the complexity of datagram-iWARP hardware solutions.

Datagram-iWARP allows for the use of a much wider variety of applications than does traditional iWARP. In particular, it allows applications utilizing datagram-based semantics to use iWARP. Our introduction of *RDMA Write-Record* [13] is, to our knowledge, the first RDMA Write technique that can work over datagram-based unreliable networks. Thus, it has the potential to bring advanced networking performance to an entirely new area of applications. RDMA Write-Record can also be of potential use on other high-speed networking architectures, such as InfiniBand.

Applications that can traditionally make use of datagram-based communication, such as streaming media or real-time financial trading applications, are part of the application set poised to make up ~90% of all Internet consumer traffic by 2014 [4]. VOIP and streaming media applications are typically built on top of protocols such as RTP [43], which can utilize UDP (datagrams) as a lower layer. Such applications have large throughput requirements, and therefore a hardware networking solution that is RDMA capable can reduce the burden on the CPU in transferring such large amounts of data to and from memory, thus freeing up the CPU for other important tasks. This solution translates into potentially reduced CPU requirements for a system, thereby providing both a power savings and a cost savings in initial costs as well as upkeep. Alternatively, this solution can result in increased system throughput, thereby increasing overall system efficiency. The use of datagrams in high-performance computing (HPC) communication middleware also can provide performance and scalability improvements for traditional scientific applications; although such applications were able to leverage RDMA capabilities using connection-based communication, RDMA Write operations over a datagram transport were previously unavailable.

It is worth mentioning that our implementation in this paper has been designed to demonstrate increased scalability and capacity of RDMA-based networks offered by performing RDMA/iWARP over unreliable transports. We also expect considerable performance improvement, especially when the solution is implemented in hardware. It is obvious that due to software incurred overheads, our implementation if iWARP over host software stack may not show its full potential for increased performance and efficiency, neither it will represent the best absolute performance numbers that can be achieved by a hardware implementation. As we will discuss in future works section, we plan for a hardware implementation of this protocol on RNET's user-programmable 10GigE NIC [42].

Arguably, the application interface could hinder the widespread adoption of datagram-iWARP. For TCP-based iWARP, the Sockets Direct Protocol (SDP) [37] can be used to provide a socket-based interface to the verbs-based iWARP stack. SDP translates traditional TCP-based networking calls such that they can utilize the iWARP set of communication verbs. The verbs networking API is not immediately compatible with traditional socket network interfaces, however, and hence a middleware layer is required. The task of rewriting applications to make use of iWARP verbs is a large and intensive one. Therefore, SDP was designed to translate sockets-based applications to use the verbs interface. SDP, however, provides support only for applications using TCP, not for datagram-based applications. In addition, since SDP is designed specifically for stream-based protocols, it is not easily adaptable to use datagram-based protocols. Rsockets [16] is a new alternative to SDP, providing a user-level sockets-compatible interface. However, Rsockets is available only for OpenFabrics stack-compatible RDMA methods, which do not support datagram RDMA Write/Read. Therefore, we have designed a socket interface that translates datagram-based networking calls to use the datagram-iWARP verbs, thus enabling datagram communication using applications to harness the speed and features of datagram-iWARP without having to rewrite any software. Unlike SDP and Rsockets, this interface is a research framework and it does not support all of the sockets API calls (for example, MSG_PEEKs) for full TCP support; but it does support datagrams, making it suitable for a subset of applications. It is designed to work between existing sockets-based applications and the software datagram-iWARP stack.

The software implementation of iWARP that was used as the code-base for datagram-iWARP originated from a project by the Ohio Supercomputer Center (OSC) [35]. The OSC iWARP project implements both user-space [7] and kernel-space [8] implementations of iWARP, with the user-space version being used for datagram-iWARP. Other software iWARP solutions have also been developed; for example, SoftiWARP, a project from IBM Zurich [29], is integrated into the OpenFabrics Enterprise Distribution stack [36]. All these projects have implemented only the traditional iWARP stack, however, and no support for datagrams or design alternatives compatible with datagrams are presented.

Work from the opposite direction, of improving existing Ethernet solutions directly without the use of a new upper-layer networking stack, has also been done. The most promising is Converged Enhanced Ethernet (CEE) [19–23,26]. CEE is essentially a collection of compatible RFCs that are meant to add advanced functionality to Ethernet networks. One of the features proposed for CEE makes an error-free transmission channel available to applications. Given the availability of this hardware-managed reliable channel, RoCE [5] is able to take advantage of the reliability to provide RDMA operations. However, the current release of RoCE does not support Layer 3 routing (IP) and therefore is applicable only to LANs. Future versions of RoCE are expected to support routing. RoCE works only over the transport modes provided with InfiniBand (no unreliable RDMA Write/Read), whereas datagram-iWARP works over unreliable transports for both channel and memory semantics (send/recv and RDMA Write/Read).

Other alternatives to iWARP have been suggested in order to improve Ethernet. Open-MX allows for Myrinet Express networking traffic to function over Ethernet hardware (MXoE) [12]. Like the local-area network (LAN) RDMA technology in MXoE, RDMA over Converged Ethernet (RoCE) [47] has been proposed as an alternative RDMA over Ethernet technology. RoCE uses InfiniBand's RDMA stack directly, using Ethernet as the link/physical layer. This has the advantage of allowing the use of all InfiniBand's basic services, including reliable connections and unreliable datagrams. Unlike our datagram-iWARP, however, the InfiniBand specifications and devices support send/recv only for unreliable datagrams. Moreover, unlike InfiniBand, wide-area network (WAN) communication support is inherent in iWARP standard.

High-speed networks have been evaluated for both 10 Gigabit Ethernet network interface cards (NICs) with offload engines [1,8,10] and others [38]. Message Passing Interface (MPI) implementations have support for multiple networks including InfiniBand, iWARP, and solutions such as RoCE [33]. MPI has been equipped with UD transport capabilities over InfiniBand for scalability purposes on large-scale clusters [28]. The MVAPICH-UD project uses an InfiniBand UD channel to reduce memory usage over the traditional RC-based channel designs. MVAPICH-Aptus [27] is a continuation of the MVAPICH-UD work and was used as the code base, which was altered to support send/recv, as well as the basis for the creation of a new transport/protocol path to enable the use of RDMA Write-Record.

Datagram-iWARP was first proposed in [39] for send/recv only. The results demonstrated runtime improvements of up to 40% over connection-based iWARP for HPC applications, particularly for applications that perform a great deal of communication. Datagram-iWARP can also provide as much as a 30% improvement in memory usage for moderately sized HPC clusters. RDMA Write-Record was first proposed and analyzed [13] in a commercial datacenter context using the iWARP socket interface. This paper is the first integrated work that provides details on both send/recv and RDMA Write iWARP network semantics for datagram-iWARP. It extends our previous work by adding MPI support for the RDMA Write-Record [30] implementation and by examining both methods in HPC and commercial datacenter contexts. It also adds support for broadcast operations over send/recv.

Our unified software implementation of datagram-iWARP for both send/recv and RDMA Write-Record has been tested on verbs-level microbenchmarks that show a maximum bandwidth improvement of 33.4% and 256% for send/recv and RDMA Write-Record, respectively, over their traditional iWARP software equivalents. Moreover, commercial application results using our socket interface show a 24.1% memory usage improvement for a Session Initiation Protocol (SIP) server application and a 43.1% performance improvement for SIP applications [11] and over 74% improvement in buffering times for streaming media (VLC [48]) applications. MPI scalability tests show up to a 30% memory usage improvement over RC iWARP. Small performance improvements can also be seen by using datagrams: 14% small-message latency reduction for send/recv with up to 14.5% and 20.7% large-message bandwidth improvement for send/recv and RDMA Write-Record, g respectively.

This paper is organized as follows. Section 2 outlines the proposed changes to the iWARP standard for datagram support for both channel (two-sided) and memory (one-sided) communication semantics. It also describes our design and implementation of iWARP over UDP. Section 3 presents our experimental results for send/recv and RDMA Write communication, for commercial data centers and HPC computing. Section 4 details our conclusions as well as future work.

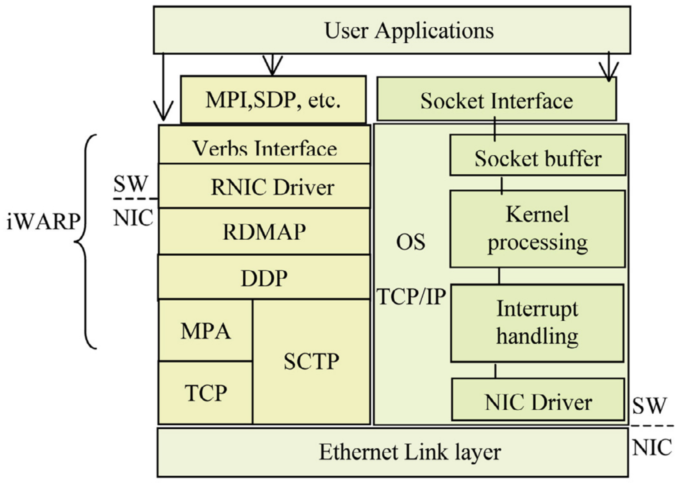

Fig. 1. iWARP standard stack compared with host-based TCP/IP.

1.1. Overview of the iWARP standard

The specifications for iWARP were first proposed in 2002 by the RDMA Consortium [40] to the Internet Engineering Task Force [25]. Utilizing a queue pair design and a communication semantic called verbs, it reflected many of the basic design choices used in Virtual Interface Architecture (VIA) [9] architectures for high-speed communication. Much like the VIA architecture, the communication stack is intended to completely bypass the operating system, avoiding extra memory copies of data as well as any context switching that would be required when utilizing the operating system (OS) communication stack. An iWARP adapter provides fully offloaded iWARP stack processing and therefore also requires fully offloaded lower-layer networking transport processing. Hence, hardware for iWARP stack processing as well as a full TOE is needed, in order to provide all the benefits of OS bypass and zero-copy operation and make iWARP hardware an RDMA-enabled network interface card, or RNIC. An overview of the iWARP stack compared with the traditional kernel communication stack is illustrated in Fig. 1.

The iWARP stack comprises three layers, with a fourth layer added for operation over a TCP transport layer. The layers are the verbs layer [17], the Remote Direct Memory Access Protocol (RDMAP) layer [41], the Direct Data Placement (DDP) layer [44], and, if operating over TCP, the Marker PDU alignment (MPA) layer [6]. The verbs layer is the direct middleware or application interface to the networking stack; it directly interfaces with the RDMAP layer.

The RDMAP layer services all the communication primitives (send, recv, RDMA Write, and RDMA Read). Verbs requests to the RDMAP layer must take the form of a work request (WR) for a given queue pair (QP). Although all requests from the RDMAP layer must be passed in order to the DDP layer, the individual WRs are processed asynchronously on the RNIC. The completion of a given WR typically results in the creation of a completion queue (CQ) entry. Some operations (e.g., send with solicited event) can trigger an event instead of creating a CQ entry, with the event being an interrupt to signal completion. Most RDMAP operations require only a single operation from the RNIC, sending data to an indicated target or, in the case of RDMA Write, a specific memory location at a given target node. Others, such as RDMA Read, require that state information about the ongoing operation be kept, since they involve a multistep operation. In the case of an RDMA Read, the initiating node requests certain data from the target node, which serves this data back to the node that initiated the request.

The DDP layer [44] is responsible for transferring data to and from the RNIC to the specified user-level buffers. The goal is to do so without any additional memory copies (zero copies from reception to buffer placement). The DDP layer achieves this through two methods: tagged (RDMA Write/Read) and untagged (send/recv).

The tagged method specifies directly in the data header where the data is to be placed, via a steering tag (STag) and offset value, and also the length of the incoming data. For this model to work, the source node must be aware of the valid areas of memory at the target node in order to create a valid message header. This requires that prior to the data transfer taking place, the nodes must exchange data on the STags and lengths of valid memory locations. Headers are checked upon arrival to ensure that the placement occurs at valid addresses only.

The untagged model does not require knowledge of valid locations in memory at the target node. Data buffers at the target nodes are handled via posted recv requests that specify the location in memory to place the data. All incoming data is matched to the posted recv requests. Unmatched (e.g., unexpected) data results in an error being passed to the ULP, since no buffer is available for reception.

The MPA layer [6] is the lowest layer of the iWARP stack. The DDP layer is message-based; therefore, in order to protect DDP messages from being unrecoverable as a result of fragmentation that may occur on the network (called middlebox fragmentation), the MPA layer was developed. This layer overcomes the middlebox fragmentation issues for stream-based transport protocols by inserting markers into the data to be transmitted, in order to point to the correct DDP header for that data, should it become fragmented. This strategy requires modifying the outgoing data to insert the markers and removing the markers from the incoming data before passing it to the DDP layer. The MPA layer is required only for operation over stream-based transports such as TCP. Message-based transports such as SCTP do not require this costly operation because they do not allow their network packets to be fragmented by middlebox network hardware.

1.2. Motivation

Current high-speed networks provide high levels of performance for reliable transports in both LAN and WAN environments. The limitations of connection-based transports previously discussed (resource usage, complexity of TOEs) represent a problem for future systems. Local computing systems will have increased node and core counts, and wide-area systems will serve more clients per node than current hardware is capable of handling. The existing iWARP RNICs are limited by the complexity of processing required for TCP streams and the behavior of TCP itself. Additional latencies caused by MPA layer handling are a result of the mismatch between message layers and stream transports. Thus, our work has two goals. First, we seek to provide RDMA functionality to a complete subset of applications (those using datagrams) that previously was not able to utilize such high-performance networking technology. Second, we seek to improve existing high-performance networks by offering a scalable (unreliable) communication option that can provide good performance while allowing for different reliability provisioning depending on individual application needs. In achieving these two goals, we also realize other significant benefits.

iWARP is an excellent high-performance networking candidate for unreliable get/put operations. The scalability of iWARP can be enhanced through the use of non-connection-based transport provisioning. While the main benefits of providing unreliable RDMA are scalability and better performance for data-loss-tolerant applications, some additional side benefits are also realizable. First, as previously discussed, the removal of the MPA layer can provide increased performance. In addition, the reduced complexity of connectionless transports also provides the benefit of easy adoption of datagram-iWARP into existing iWARP silicon. Alternatively, datagram-iWARP provides the opportunity for a datagram offloaded iWARP solution while offering onloaded TCP processing. Such a solution would be much less expensive to produce, given its lack of a TOE and MPA processing, and hence would be potentially attractive for datacenter users that may not require reliable iWARP functionality, where applications that can tolerate loss can benefit the most from our proposed extensions. Furthermore, datagram-iWARP provides the opportunity to support broadcast and multicast operations that could be useful in a variety of applications including media streaming and HPC for collective operations. Since the iWARP RNIC is compatible with the OpenFabrics verbs API [36], methods developed for iWARP can also be easily adapted for use with other verbs-compatible interfaces such as InfiniBand. iWARP's WAN support is also an important feature for leveraging datagram support, since many datagram-based applications require WAN capabilities.

In the case of packet loss, TCP and SCTP both guarantee in-order delivery of data. Therefore, network jitter is experienced because there is a delay in delivering data that has already arrived but is blocked by an incomplete message at the head of the receive queue. For a datagram transport such as UDP, data is delivered as it arrives, and this approach helps reduce network jitter, particularly for time-dependent data transmissions. Applications that can tolerate packet loss can make use of datagram-iWARP, enhancing their performance by using OS bypass, offloading, and zero-copy techniques. These performance-enhancing networking features are unavailable for such applications in traditional iWARP.

Scalability is a prime concern of high-performance networks. The communication channel isolation of connection-based transports is excellent for flow control and reliability of individual data streams, but it limits the resource sharing that can occur between the connections. Connections must have a current state, and communication over a connection can occur only to a single destination point. The resource requirements per connection are further increased through the design of some HPC middleware. Some MPI implementations prepost receive buffers in support of Eager communication protocols on a per connection basis. This preposting is beneficial to performance because memory is preregistered, but it uses memory for every connection. Although some schemes such as shared receive queues have been designed to reduce connection-based memory requirements in these environments, they are still not as efficient as a single datagram receiver of data. Datagram transports do not have to keep state information as a connection does. For connection-based iWARP implementations, state information is kept in local memory on the RNIC. Alternatively it can be kept in system memory, in which case it must be accessed through slower requests over the system buses.

Reducing the complexity of the networking protocols through the use of UDP will help reduce message latency and close the latency gap between iWARP and other high-speed interconnects. Offloading the network processing onto the RNIC will reduce the CPU requirements of providing high-throughput traffic for datagram applications by lessening the data movement responsibilities of the CPU. This will enable businesses to concentrate their infrastructure investments directly into networking technologies and capacity, thereby reducing the amount spent on CPU hardware while still supporting high bandwidth and low latency. This is a common benefit of WAN-capable RNICs that can support both WAN and LAN-based applications. Another performance advantage of using datagram-iWARP is the existence of message boundaries, as discussed in Section 1. SCTP does allow message boundaries like UDP; however, it provides even more features than those in TCP and consequently is more complicated. SCTP is also not as mature as either TCP or UDP and therefore does not have as much application support nor as long a history of performance tuning as does either TCP or UDP.

The elements of datagram-iWARP design that improve performance will have minimal impact on the implementation cost. Adding datagram-iWARP functionality to existing iWARP RNICs would be relatively easy and inexpensive, and the most likely form in which datagram-iWARP can be leveraged for real devices.

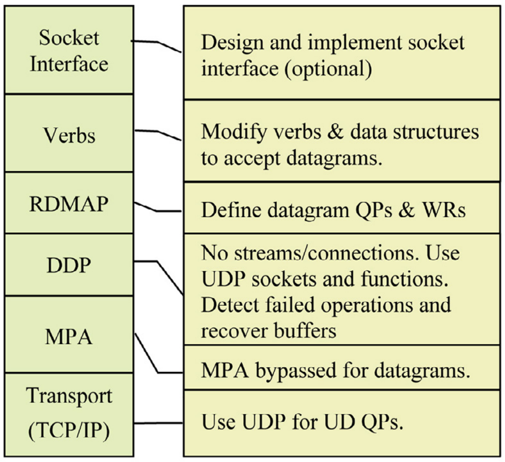
Socket Interface
Verbs
RDMAP
DDP
MPA
Transport (TCP/IP)
Design and implement socket interface (optional)
Modify verbs & data structures to accept datagrams.
Define datagram QPs & WRs
No streams/connections. Use UDP sockets and functions. Detect failed operations and recover buffers
MPA bypassed for datagrams.
Use UDP for UD QPs.

Fig. 2. Changes for datagram-Iwarp.

2. Methodology

Datagram-iWARP represents a significant shift in the overall design of iWARP, since the current standard is based entirely on reliable connection-based transports that provide ordered delivery. Because datagram-iWARP is designed for an unreliable, non-ordered network fabric, many of the existing assumptions made in the iWARP standard regarding the data delivered by the lower network layers are no longer applicable. This section first outlines the changes needed to enable both two-sided (channel-based) and one-sided (memory-based) methods of communication, including changes to the requirements in the iWARP standard, as well as required and optional changes to the behavior of iWARP. It then outlines the design of RDMA Write-Record and discusses how its behavior differs significantly from that of the traditional iWARP RDMA write operation.

2.1. iWARP design changes

Given a connectionless traffic flow, the source address and port of the incoming data must be reported to the application receiving the data. This process requires that existing verbs be altered in order to pass along the required data structures for datagram-based traffic. Alternatively, separate datagram-based verbs can be introduced to handle datagram traffic. Existing compatible verbs interfaces such as the OpenFabrics (OF) [36] verbs specification can allow for datagram-based traffic (as send/recv datagram traffic is supported in other OF verbs compatible interconnects such as InfiniBand). Therefore, the iWARP-specific verbs to handle datagram traffic can also be designed to be compatible with OF verbs through a verbs-to-verbs interface. An example of the additional verbs that support datagram traffic is a datagram-compatible post_send verb and a post_recv verb. Additional changes to verbs for polling the completion queue are also required, as are changes to verbs for creating and modifying queue pairs for datagram-based communication. A high-level overview of the changes required for datagram-iWARP is shown in Fig. 2.

Since datagram QPs require different initialization information from that of connected QPs, the QP creation/modification verbs need different inputs from those of existing connection-based verbs. The modification of datagram QPs also requires changes in the requirements for the inputs. For example, datagram QPs cannot be set up on the fly as data is delivered for transmission. Therefore, datagram QPs need to allow for a basic socket allocation and communication setup to occur when the QP is first created. This creates a valid socket that can then be used to transition the QP into the ready-to-send (RTS) state.

Communication transmission verbs (send and RDMA Write-Record) must be altered so that they receive a valid destination (IP address) and port with every operation, since they cannot rely on past behavior to determine the destination for a given data transmission. For connection-based communication, this is not an issue because data over a given connection always flows to the same destination. Likewise, the verbs for receiving data must be altered such that they deliver the address and port of the source of the transmission. For recv, this alteration is relatively straightforward, through the addition of data structures to relay this information. However, this is not done in the recv verb itself, but rather as a common change to both recv and RDMA Write-Record, by reporting the source address and port in the completion queue element passed back to the requesting process after a completion queue poll request.

2.1.1. iWARP standard definitions

In the current iWARP standard, a work request is completed as soon as its delivery can be assured. For a reliable connection, this assurance occurs upon passing the data to the underlying transport layer. For an unreliable transport, such an assurance can never be provided. Therefore, for datagram-iWARP, a work request completion occurs when the data is passed to the LLP for delivery, without the expectation that it is guaranteed to be delivered.

The RDMAP specifications in the iWARP standard [17] Section 5.1 states that LLPs must provide reliable in-order delivery of messages. The DDP standard [44], Section 5, item 3, states that the LLP must reliably deliver all packets. These specifications must be changed to allow for the operation of datagram-iWARP.

The DDP standard, Section 5, item 8, states that errors that occur on an LLP stream must result in the stream's being marked as erroneous and that further traffic over the marked stream is not permitted. Likewise, the RDMAP specification requires an abortive teardown of an entire communication stream should an error occur on that stream. These requirements must be relaxed for datagram-iWARP, where for the case of a datagram QP, the error must be reported to the upper layer, but no teardown occurs and traffic can continue to traverse across the QP. This approach is necessary because the QP might be communicating with multiple targets or receiving data from multiple sources. Also, an error occurring over an unreliable transport might be recoverable.

2.1.2. *iWARP behavioral adaptations*

The iWARP standard has been designed for an LLP that offers guaranteed in-order delivery of data. For datagram-iWARP we must adapt the behavior of the communication stack such that it can be compatible with an unreliable transport. In many cases, the applications utilizing a UD iWARP must be aware of its unreliable nature and have provisions to deal with data loss and reordering. This requirement is true, for example, for streaming media applications and even some implementations of HPC application middleware such as MPI. However, for the datagram-iWARP specification over a reliable datagram (RD) transport, one that provides both delivery and order guarantees, many of these behavioral adaptations are not required. Therefore, unless expressly stated, the proposals in this section refer to those for iWARP operating over an unreliable transport.

The lack of delivery guarantees for datagram-iWARP requires that the polling for the completion of a recv operation as well as the completion of an RDMA Write-Record operation have a timeout period associated with the polling request. Otherwise, an infinite polling loop may result, which would cause an application to fail.

Connection establishment and teardown procedures are not required for datagram-iWARP. For datagram-iWARP, a QP can be transitioned into an RTS state after a QP is created and an LLP socket is assigned to the QP. Since this setup can be accomplished without any transmissions to the target of the datagram QP, the other parameters used for the QP, such as the use of a cyclic redundancy check (CRC), must be configured by the ULP. Doing so also eliminates the requirement that the ULP configure both sides of the QP (target and source) at the same time.

Since the LLP may or may not handle reliable service and order guarantees, any requirements for state information on underlying LLP pseudo-connections are the responsibility of the LLP and not of iWARP. Therefore, any associated pseudo-connection establishment, modification, or termination that is required must be the responsibility of the LLP.

Error management behavior must be changed in the iWARP stack. For datagram-iWARP, errors must be tolerated, by reporting errors as they happen to the upper layer, but not causing a complete teardown of the QP. Consequently, the application may decide on the course of action to take upon notification that a communication error has occurred. If one is using an RD transport, the error is passed along to the application in the same manner as for a UD LLP, but the QP is placed into an error state. The error state prevents any further traffic on the QP until the error is dealt with by the application by resetting the QP state into the RTS state. This approach also requires that an error message detailing the error (and, in the case of a RDMA Write or Read communication, the message sequence number) be sent to the message source. The error message is locally placed in an error queue as opposed to the termination queue used in traditional iWARP.

Message segmentation and the requirement for marking of messages are significantly altered for datagram-iWARP. Since the proposed LLP, UDP, has a maximum message size and since UDP delivers the message in its entirety, message segmentation is not needed in the iWARP stack. The application layer is responsible for ensuring that data transfers larger than 64 kB are segmented and reassembled properly. The MPA layer and its marking of packets in order to facilitate reassembly are also not needed. Since datagram packets once segmented into maximum transmission unit (MTU)-sized frames by the IP layer are not permitted to be further segmented by network hardware along the transmission route (unlike TCP traffic), there is no need to perform the costly activity of marking the packets so they can be reassembled correctly at the target system.

2.1.3. *Optional iWARP changes*

Datagram-iWARP always requires the use of a CRC when sending messages, unlike TCP based iWARP in which it is negotiated whether or not to use iWARP CRC when first connecting to a peer. This requirement allows for the creation of datagram QPs without any communication between the source and target nodes (as there is nothing in need of negotiation before communication), as well as ensuring that no conflicts occur in terms of CRC use over a datagram QP, given that it can communicate with several other systems. Since the proposed LLP for the UD transport, UDP, does not require the use of a checksum, and its checksum is inferior to a CRC, it must be performed in the iWARP stack such that data transmission errors are identifiable.

2.2. *RDMA Write-Record*

Existing RDMA (one-sided) operation definitions do not allow for operation over unreliable datagrams. Because of its sender-oriented nature, the target node is not aware that an incoming transmission may be occurring. Over a network that provides guaranteed in-order delivery, the source can be assured that data passed to the lower layer networking protocols will arrive properly at the target node. Such a guarantee cannot be made for an unreliable datagram-based transport. Current specifications send a notification send/recv message after an RDMA Write operation has completed, in order to notify the target node that the data has been successfully transmitted into its local memory [40]. An optimization of this basic method allows the target node to determine whether the data is valid, not by an additional message, but by the modification of a given bit in memory once the operation is complete. Both methods rely on a reliable transport. With an unreliable transport, the notification message or the message segment containing the notification bit could be dropped. Alternatively, the notification message could arrive, but the message itself might not; this situation would cause significant problems because the memory region would not contain correct data and most likely would cause application failure or invalidation of application results. Consequently, the traditional RDMA model does not adapt well to an unreliable network environment.

Therefore, to facilitate operational RDMA over unreliable datagrams, we propose a new method called *RDMA Write-Record*. RDMA Write-Record uses the iWARP-tagged model data structure while changing the behavior at the source and target nodes. At the source, the RDMA operation is not followed by any additional notification operations in order to complete the operation at the target node. It sends the data to the target and cannot assume that the data was received. If such information is required, it must be supplied by using a reliable LLP or by notification messages at an upper layer. The behavior at the target node changes significantly. Upon receiving an RDMA message, the target node places it in memory as it would for a connection-based message; but for a datagram message it also records that the memory write took place, the location of the transaction, and the length. This information is then accessible through the completion queue. Hardware may be capable of aggregating many independent datagram layer messages (UDP messages) into a single CQ entry. For the software implementation of RDMA Write-Record, each UDP message gets a single CQ entry and for messages larger than UDP MTUs (64K) the software coalesces the incoming packets into a single CQ entry (message). This CQ entry indicates which UDP messages (64K chunks and the <64K tail of the message) of the larger iWARP message completed successfully. The lifetime of data for a message is determined using a timeout in between the UDP messages making up a larger iWARP message. If this timeout expires with no further message reception, the resulting incomplete message completion is placed on the CQ. The determination of the timeout period is implementation and situation specific, where a short enough timeout for performance is desirable, without missing packets that may be delayed. If UDP messages arrive after the timeout has occurred, the messages are dropped. Further details on packet loss will be discussed in Section 3.2.3, but use of iWARP in an intra-datacenter context results in very few packet losses and low latency, which can provide very short timeouts and high performance. Since this method uses existing data structures and queues already present in traditional iWARP, its implementation is relatively lightweight. The design is illustrated in Fig. 3 and compared with a traditional RDMA Write operation. This design also has the benefit that it is easily compatible with socket semantics and makes an interface for socket-based applications easier to implement. The reason is that a completion notification exists for an RDMA operation, so it can be easily polled for via the completion queue and the requisite information passed back to the polling application.

For further adaptability, two possible methods exist for creating CQ entries for RDMA Write-Record operations that allow applications to be designed with either a message-based networking semantic or a memory-based networking semantic. In the case of the message-based semantics, each RDMA Write-Record operation is entered as an individual entry in the completion queue. Therefore, when the application polls for a completion, each completion entry can be interpreted as a separate message received. This compatibility is essential to making RDMA Write-Record applicable to existing message semantic-based applications such as those utilizing sockets interface-based networking.

The alternative paradigm is memory-based semantics, in which there is no need to conceptualize network messages but instead regard transfers as simply memory writes and reads. In this paradigm, we need to know what areas of memory are valid, but notification of individual complete memory operations is unnecessary. The CQ entry method that is used for the message-based semantics approach is still valid for this approach (thereby reducing overall device complexity), since we can create a validity map for any requesting application by simply traversing the CQ and using the CQ elements to translate the message completions into memory ranges for a single validity map structure. This approach provides additional application interface flexibility while only marginally increasing device complexity and does not require a QP to be declared as using only one of the two methods. For the software implementation of RDMA Write record, the granularity of the memory validity map will be rather coarse, at typically 64KB UDP message size granularities (with the exception of the tail of large message that may be less than 64K in size). This means that the memory semantic approach is similar to the message based approach except for the presentation of the missing regions (one based on total number of messages, the other explicitly outlining the memory region that is invalid). For a hardware implementation, the granularity can be at an individual UDP packet size instead of at message granularity, meaning that a granularity of approximately 1472 bytes is possible (1500 byte MTU – 8 byte UDP header – 20byte IP header).

Implementation wise, maps can be part of the registered memory windows with map granularity at 1 bit per 64B (typical cache line size). In every write to the remote memory, such structures (at the host or the NIC) can be updated based on data-sink side completion notifications. Every update of the validity map inside a registered memory window can mark a portion of that window as valid or invalid, and every read will clear those markings. Obviously if a packet loss goes undetected (e.g., an entire message is lost), no memory areas are marked. But in the case of a partial loss, the lost portions are marked as invalid. Since validity maps can be assembled from multiple low-level CQ entries, they are as bounded as CQ entries.

It is worth mentioning that in RDMA (UD or RC), a separate mechanism is always required to make sure the data sink application reads the data before an overwrite by a subsequent RDMA write (record). Moreover, note that RDMA is one sided, and RDMA write record is preserving that nature, therefore, completion at the data sink is primarily for data validity (and in fact such information can be used instead of polling on actual data to realize a new data arrival).

The manner in which completions are to be reported can be specified in the call to the CQ polling verb. We note that this method does not remove the requirement for synchronization between communicators before the target memory is reused. The state of the memory may be undefined if multiple writes are performed to a single memory location without synchronization.

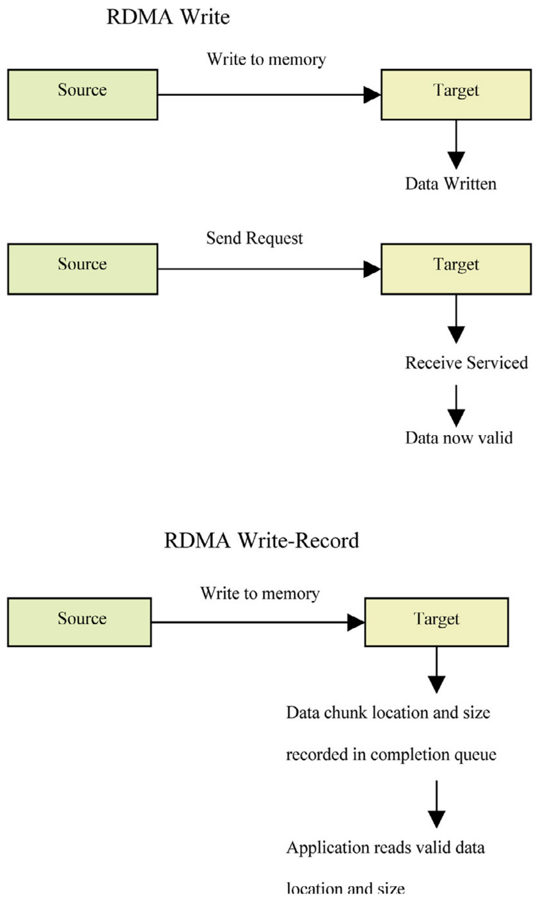

Fig. 3. Comparison of RDMA Write over RC and RDMA Write-Record over UD.

However, such synchronization is out of the scope of the iWARP as the underlying transport fabric and needs to be addressed through application/middleware layers. For example, in an HPC setting, where sharing of such memory windows is common for one-sided operations, middleware such as MPI may provide *adequate* reliability and synchronization, so that using unreliable iWARP for scalability and performance is desirable.

Our RDMA Write-Record design differs from existing send/recv operations. In the case of send with solicited event, a recv must be posted before the specified event can be triggered upon reception of the data; this process also triggers an event to occur immediately upon reception (an interrupt). RDMA Write-Record, on the contrary, reports that an operation has occurred only when the upper layer software requests information concerning completed operations. RDMA Write-Record also varies from the InfiniBand RDMA Write with immediate operation in much the same way, in that it requires that a recv be posted in order to receive the immediate data. Therefore, RDMA Write-Record is a truly one-sided operation as opposed to send with solicited event and RDMA Write with immediate operation, which both require a posted recv operation at the target in order to function and hence are two-sided operations.

2.2.1. *Packet loss design considerations*

The memory-semantic-based RDMA Write-Record paradigm means that requiring that all operations be less than the maximum data size for a datagram (64 kB) is not necessarily fair. Therefore, we have designed support for messages larger than 64 kB, as well as support for partial-message placement. For networks that are relatively error free, this can lead to an increase in performance because applications can pass large messages to the iWARP stack without first segmenting them into 64 kB messages. Segmentation of even 64 kB messages occurs over traditional Ethernet networks, which operate over a 1500-byte MTU. UDP relies on the IP layer for segmenting a 64 kB message into several MTU-compatible messages and reassembling that message at the target node. Messages larger than 64 kB cannot be handled by this mechanism in UDP and therefore must be segmented at the iWARP level before being passed to UDP. This approach results in several smaller message segments in iWARP that must be reassembled at the target node into a single larger iWARP message. The partial-message placement support added to iWARP for RDMA Write-Record refers to this level of message segmentation, 64 kB iWARP message segments. It allows for the

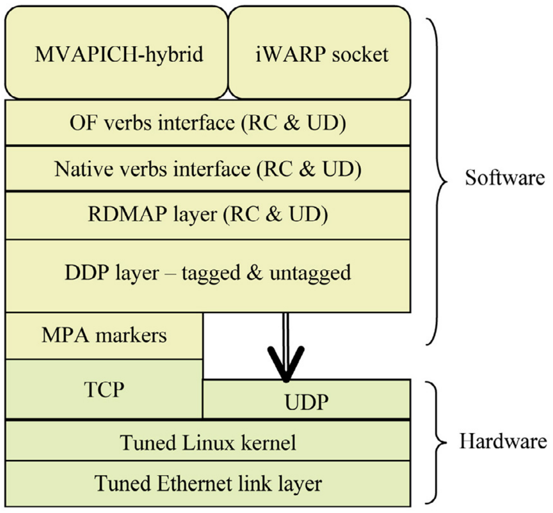
MVAPICH-hybrid
iWARP socket
OF verbs interface (RC & UD)
Native verbs interface (RC & UD)
RDMAP layer (RC & UD)
DDP layer – tagged & untagged
MPA markers
TCP
UDP
Tuned Linux kernel
Tuned Ethernet link layer
Software
Hardware

Fig. 4. Software implementation of datagram-iWARP.

received portions of a message to be placed into memory as they arrive; the resulting memory validity map reflects the missing portions of the overall larger message. This approach is appropriate for applications that can handle some packet loss, such as online gaming, VOIP, and streaming media applications that also can make use of large messages. Other applications such as streaming video applications can also handle packet loss as well as invalid data and can make use of partial-message placement.

Packet loss for large, multisegment datagram messages would normally result in complete message loss, and for many applications this is still the case. Partial-message delivery is not a requirement of RDMA Write-Record. It is detailed here for the subset of applications that can handle packet loss, as well as for future Ethernet networks, such as the changes proposed for CEE [5], where error-free streams for datagrams can be provided. Such network channels could make use of iWARP hardware segmentation and reassembly of large messages.

2.3. Datagram-iWARP software implementation

To evaluate the proposed datagram extension to the iWARP standard, we have developed a software implementation of datagram-iWARP based on a TCP-based iWARP implementation from the Ohio Supercomputer Center [35]. This allows for a direct comparison between the UD and RC modes of datagram iWARP, using a publicly available RC implementation. Fig. 4 shows an overview of this implementation including the additional upper-layer interfaces we have added—namely, the iWARP socket interface and compatible MPI layers—as well as modification to the existing OF verbs interface in order to make it compatible with more OF verbs middleware and applications. The changes required to the verbs, RDMAP, and DDP layers, as described in Section 2.1, are reflected in the layers of the stack indicating both UD and RC support. While this implementation is fully capable of operating over a reliable UDP transport, all the testing and results of the implementation were performed over an unreliable UDP.

Modifications to the existing incomplete OF verbs interface on top of the native software iWARP verbs was necessary because of the base RC iWARP implementation used for the design. OF verbs were originally designed for InfiniBand (OpenIB verbs) but are now also used to support iWARP hardware in a unified driver. Since the RC implementation uses native (non-OpenFabrics) iWARP verbs, they must be translated over to OF verbs for use at the MPI layer. An existing MPI implementation [27] was adapted to use the OF verbs interface of our iWARP stack. A socket interface was also added in order to facilitate the use of socket-based applications without needing to rewrite the existing networking code.

Our datagram-iWARP implementation uses CRC error checking at the lower DDP layer for datagrams to ensure correct reception of individual packets. Checksumming may be performed at the UDP layer and is unnecessary given the CRC check at the DDP layer. Therefore, it is recommended that checksumming be disabled at the UDP layer to further enhance performance. The implementation uses round-robin polling over sockets to ensure fair service to all QPs in the software RNIC. For improved performance, it has been enhanced to use I/O vectors for UDP communication (similar to TCP) to avoid extra sender-side and receiver-side copies. I/O vectors enable the gathering/scattering of data from noncontiguous memory locations in a single operation, avoiding any intermediate memory copies when assembling a datagram with its header and associated data payload. Our implementation also avoids segmentation of DDP messages into MTU-size datagrams, using the available Ethernet hardware segmentation provisions.

The datagram-iWARP implementation has been extended to allow for IP-level broadcast operations. Using the existing IP broadcast provisions available for Ethernet networks, we adapted the datagram-iWARP implementation to demonstrate the bandwidth achievable using IP-level broadcasting. Send/recv is supported for broadcast operations, as the receiver managed data placement is useful for assuring correct delivery to multiple target nodes. RDMA Write-Record is not an appropriate solution for IP broadcast, as the location to store the incoming data in memory will not be identical on each of the target nodes. If such a guarantee were possible, RMDA Write-Record could be used for IP broadcasting. The addition of broadcast support demonstrates that datagram-iWARP can be applied to multicast as well as IPv6 broadcast/multicast with ease.

2.4. iWARP socket interface

The compatibility of iWARP with existing software without the need for a long and expensive networking code rewrite is a desirable feature. Existing iWARP implementations can operate over the Socket Direct Protocol or the rsockets interface, which allows sockets-based applications (nondatagram) to run over iWARP verbs. However, SDP was designed specifically for TCP-based applications and does not support datagram-based applications. Rsockets supports both TCP and (very recently) UDP but does not support the use of RMDA Write-Record. Therefore, an interface was needed to allow datagram-based applications to take advantage of datagram-iWARP without having to rewrite networking code in order to use verbs. A lightweight interface was developed to facilitate both RC-based and UD-based socket networking code to work with iWARP. This interface is not as complex or full featured as SDP or rsockets but provides a baseline with which the performance of RC and UD socket-based applications can be compared.

The iWARP socket interface is loaded in the same manner as is SDP/rsockets. When running an application, the interface is dynamically preloaded and uses networking calls that override the existing operating system network calls, passing them to the iWARP networking stack. The iWARP socket interface does not fully override all networking functions. The existing socket creation and modification operations are still performed by the operating system, since the iWARP software interface directly uses OS-level sockets. The interface also does not support all operation configurations for sockets but provides enough operation to work over several commercial applications. Datagram-iWARP hardware would require a different socket interface model in order to fully leverage OS bypass, one that would handle socket creation for the RNIC as well as provide for all possible socket configurations to enable universal compatibility. A socket interface for hardware could enhance performance by offering both a buffered-copy and zero-copy transfer method with threshold switching at specified message sizes to further leverage the capabilities of both send/recv and RDMA Write-Record.

The iWARP socket interface operates by allowing for both TCP and UDP-based iWARP sockets to be opened, using the relevant iWARP lower-layer protocol. It limits sockets to having only one QP associated with them, so that socket-based applications do not have multiple streams over a single socket.

The interface determines the type of socket issuing a request and uses either RC or UD for the socket type. The socket data structure passed with each request stores the destination address, source address, and port; the interface stores a QP to file descriptor (socket number) mapping in order to track initialized iWARP sockets. If a socket passed to the interface has not been previously initialized to work over iWARP, then the appropriate steps are taken to create an iWARP QP over the socket.

This interface is suitable for determining the performance of datagram-iWARP with some popular socket-based applications. However, it is not a comprehensive alteration of the existing SDP standard in order to adapt it specifically to use a datagram semantic. Altering SDP would have the positive benefits of hiding the socket creation from the applications and is essential for a hardware implementation to separate the socket-based networking semantic at the application level from the verbs-based semantics of the hardware itself. Creating a datagram SDP-equivalent protocol would be complex and require major changes or additions to the existing standard. However, such an interface without compatible hardware would not allow for OS bypass, limiting its usefulness. Our goal with the iWARP socket interface is to demonstrate that a datagram socket to verbs translation is possible and to demonstrate the potential benefits that datagram-iWARP could bring to existing datagram sockets-based applications.

2.5. MPI over datagram-Iwarp

Although the primary focus of datagram-Iwarp is to allow for improved performance on applications that can tolerate data loss, the use of datagrams could also potentially provide good scalability to MPI applications. To evaluate the scalability of datagram-based Iwarp for HPC applications, we have adapted MVAPICH on top of OF verbs over our software datagram-Iwarp implementation. The hybrid communication channel from the MVAPICH-Aptus over InfiniBand implementation [27] has been used for this development. MVAPICH-Aptus is an implementation of MPI that enables the use of both UD and RC traffic. Originally designed for InfiniBand UD and RC transports, the implementation was intended to create a more scalable MPI solution by offering both high-performance RC channels and low-resource-requirement UD channels. The design is intended to allow for only a given threshold of RC channels to be created while UD can form the majority of communication channels. The goal is to create scalability by limiting the total number of RC QPs. Since the UD channel offers no reliability guarantees, a lightweight reliability mechanism incorporating sequence numbers, acknowledgments, and timeouts was added.

In order to adapt MVAPICH-Aptus to use our datagram-Iwarp implementation, the following modifications were required for both send/recv and RDMA Write-Record.

1. MVAPICH UD-based connection management was modified to make it compatible with datagram-Iwarp: This modification required creating and initializing datagram sockets at the UDP level to pass to datagram Iwarp.

2. MVAPICH RC-based connection management needed to be modified because the MVAPICH hybrid channel used on-demand connection management [27]. Since MVAPICH-Aptus was designed for InfiniBand-based RC, semantic differences between TCP and InfiniBand RC needed to be resolved by modifying the handshaking steps for MVAPICH connection establishment. The handshaking was modified to include required information for Iwarp and to perform the socket connections at the end of the handshaking.
3. Parts of the code supporting features or operations not needed or supported by our Iwarp implementation were removed, such as immediate data, GRH headers, shared receive queues, eXtended Reliable Connections, service levels, LIDs, and LID mask control.

RDMA Write-Record functions in a different manner from traditional RDMA Write. Therefore, existing support for RDMA Write operations needed to be supplemented with a new transport channel/protocol capable of supporting RDMA Write-Record. The resulting UD-Write communication mechanism incorporates the functions of the existing RDMA Write and UD channels. RDMA Write-Record uses reliability mechanisms while utilizing a rendezvous protocol for buffer management between nodes. The behavior of the target node is also slightly changed because the RDMA Write-Record channel can complete operations when the data arrives.

3. Results and discussion

In this section, we briefly describe the experimental setup. We then present the experimental results and our analysis.

3.1. Experimental platform

We use two clusters for our experiments. Cluster C1 is a set of four nodes, each with two quad-core 2 GHz AMD Opteron processors, 8 GB RAM, and a NetEffect 10-Gigabit Ethernet (10GE) card connected through a Fujitsu 10GE switch. The OS is Fedora 12 (kernel 2.6.31). Cluster C2 contains 16 nodes, each with two dual-core 2.8 GHz Opteron processors, 4 GB RAM, and a Myricom 10GE adapter connected to a Fulcrum 10GE switch. The OS is Ubuntu with kernel version 2.6.27. The 10-Gigabit Ethernet NIC is used only in Ethernet mode, supporting the software iWARP stacks for the results. The iWARP RDMA features of the NIC were not used so that a comparison could be made between TCP and UDP iWARP. All tests were run using a software iWARP implementation over sockets on Ethernet hardware.

3.2. Microbenchmark performance results

Microbenchmark results are presented for both send/recv and RDMA Write-Record as compared with the connection-based traditional iWARP operations. The microbenchmark tests were performed on the C1 cluster using one pair of nodes. All tests were run and averaged over at least 10 benchmark runs, with each run comprising an average of thousands of iterations, for each message size.

3.2.1. Verbs-layer microbenchmarks

For the verbs level, the latency results for native iWARP verbs are shown in Fig. 5. Native verbs for iWARP are the native TCP software verbs without the OFED compatibility wrapper. The figure combines the ping-pong latencies for send/recv, RC RDMA Write, and UD RDMA Write-Record. The lowest latencies are for UD send/recv and UD RDMA Write-Record, 27–28 μs for messages less than 128 bytes. These latencies are much higher than existing hardware implementations of iWARP (being in the 1–3 μs range for small messages [3]), because the use of a software implementation of the UDP/TCP stacks instead of using the ToE adds significant latency through kernel processing. Latencies of ∼30 μs are reasonable for a typical sockets operation through the Linux kernel for our particular kernel and hardware generation. Therefore, the overhead of software processing from the iWARP implementation itself is minimal, as most of the difference in performance comes from the lack of OS bypass capabilities made possible through iWARP hardware. Therefore, for the interpretation of the results of the software implementation and its carry over to impact on potential datagram-iWARP hardware, latencies improvements should be thought of in terms of percentage improvement, not absolute values.

Datagram-iWARP is consistently better than both RC send/recv and RC RDMA Write by 5–6 μs for small messages. In terms of magnitude, this seems like a large difference, but when translated down to hardware level latencies (approximately one order of magnitude) this results in an expected improvement of 500–600 ns in latency. This direct benefit can be attributed to the efficiencies in the datagram-iWARP stack, primarily the lowered communication overhead and lack of costly packet marking in the MPA layer. The cost of the MPA marking in terms of overall impact on the iWARP software stack is approximately 18% of the total code in the stack. As such removing the requirement for such a significant portion of the code path is significant in terms of latency improvements. Lowered overhead partially results from the lack of reliability-related communication and TCP-based flow control. TCP protocol processing can be resource intensive and adds to the overall latency of packet delivery, resulting in a reduction of the overall system capacity in processing packet flows. Since the microbenchmarks were performed on a local-area network with relatively error-free operation, the advantages of lower reliability requirements are fully realized. The performance improvements shown in this section are consistent with the code improvements from eliminating the MPA layer and utilizing UDP over TCP, which is well known to have latency advantages over TCP [31]. There are differences between software-based

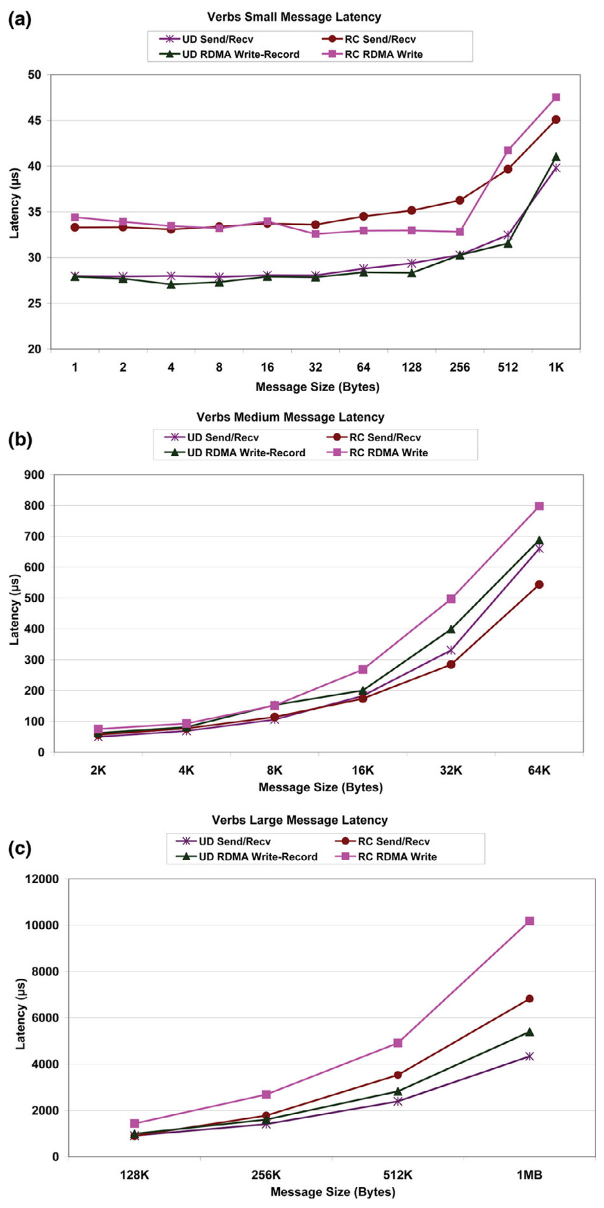

Fig. 5. UD iWARP vs. RC iWARP native verbs latency.

networking stack processing and hardware offload versions. However, the overall percentage improvement in latencies and throughput shown for the iWARP software implementation represent the reductions in network stack processing complexity, which while different in magnitude when applied in hardware are still representative of expected hardware performance improvement.

All of the verbs-level micro-benchmarks were performed using many trials such that the results are within 0.5% with 99% confidence 19 times out of 20. For message sizes up to 2 kB, UD send/recv has an 18.1% average improvement over RC send/recv. UD RDMA Write-Record offers a 24.4% average improvement over RC RDMA Write. RC send/recv shows superior performance between 16 kB and 64 kB message sizes but is overtaken by both UD methods for messages of 128 kB or larger. Jumbo frames force the UD approaches to process/send two packets, which results in additional overhead versus the TCP based stream with a hardware stateless offload. For the RC RDMA Write case, there is overhead associated with doing this in software. The UD overhead becomes less of an issue with larger message sizes and the RC related overhead in the iWARP library results in a reverse of this result for larger message sizes. The 64 kB message size is the point at which we exceed the maximum size of a datagram message and therefore must segment the message into 64 kB chunks before handing them over to the LLP.

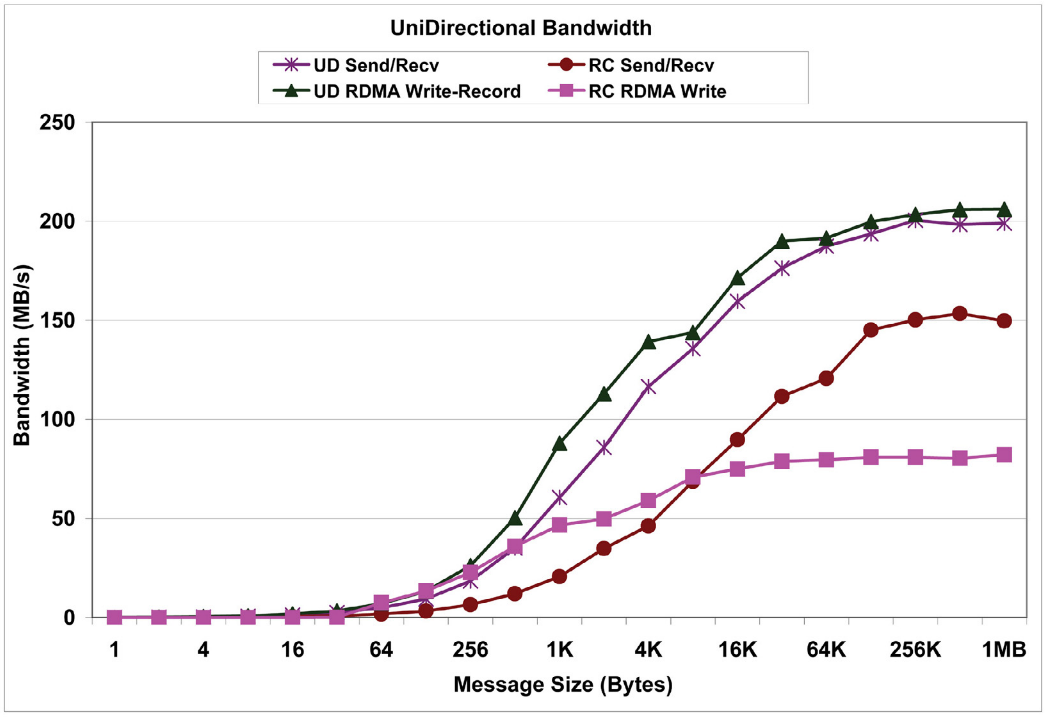

Fig. 6. Unidirectional verbs bandwidth.

Unidirectional bandwidths of the different iWARP methods are illustrated in Fig. 6. Both UD-based methods outperform their RC equivalents by a wide margin. This result is particularly observable at larger message sizes, where UD send/recv shows its maximum improvement of 33.4% over RC send/recv at 256 kB messages and where RDMA Write-Record has a 256% advantage at 512 kB over RC RDMA Write.

Although the performance of UD send/recv and UD RDMA Write-Record does not differ significantly, the performance of the different methods at the underlying network MTU of ~1500 bytes is of interest because many datagram-based applications use messages of this size. For 1 kB messages, UD RDMA Write-Record has a bandwidth of 188.8% higher than RC RDMA Write, and UD send/recv has a maximum bandwidth 193% higher than RC send/recv. UD RDMA Write-Record also enjoys an advantage of 31% over UD send/recv at 1 kB message sizes. These results can be attributed to the lowered processing overhead at the transport layer (when using UDP), as opposed to TCP as the underlying transport. The most important differences between UDP and TCP that are creating these large performance gaps are the lack of congestion control and reliability in UDP. In particular due to these tests using a single UDP send/recv, UDP is able to take advantage of all of the available bandwidth without concerns of congestion encountered from other traffic. In practice, some application will require reliability from the network transport, and utilizing reliable UDP instead of UD will result in less bandwidth improvement than seen with pure UD. However, for applications that can deal with data loss and require high bandwidth low latency transmissions, this approach can have significant benefits.

3.2.2. Send/recv broadcast

A verbs-level benchmark was developed to test the bandwidth achievable over a network using IP-level hardware broadcasting operations. A source node was tasked with delivering data to multiple target nodes via IP broadcasting provisions. The bandwidth is expected to scale well with an increasing number of target nodes. With no overhead, the bandwidth should scale linearly. The bandwidths for different numbers of target nodes are shown in Fig. 7. One can observe that for two and three receivers, the bandwidth scales well for small and medium messages. Greater overhead is observed for copying large messages by the broadcasting switch, leading to nonideal scaling. The overhead is approximately 12–13% for messages greater than 64 kB with three receivers. This overhead is not excessive, and is preferable to having the software iWARP stack replicate packets and send them to multiple targets. A slight change also occurs in the slope of the bandwidth curve at the 8 kB message size. At this message size, the network MTU is exceeded, and segmentation of the message into multiple packets at the IP level must occur. This extra processing causes a slight drop in the efficiency of the networking operations.

3.2.3. Packet loss and performance

The results shown thus far have illustrated the performance of datagram-iWARP in LAN conditions where little packet loss occurs. To study datagram-iWARP in a WAN environment, one must consider its verbs-level performance in the presence of packet loss. Linux provides traffic control systems that allow for packets entering the outgoing traffic queue to be dropped in a

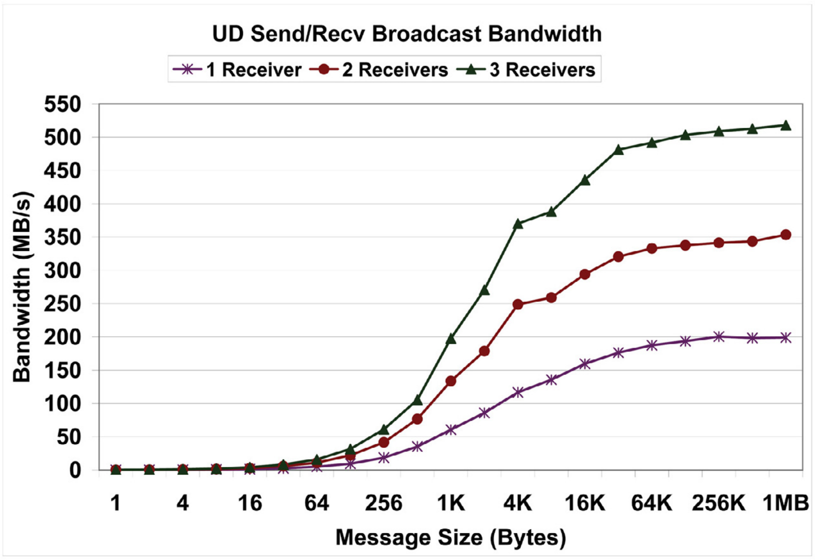

Fig. 7. Send/recv broadcast aggregate bandwidth.

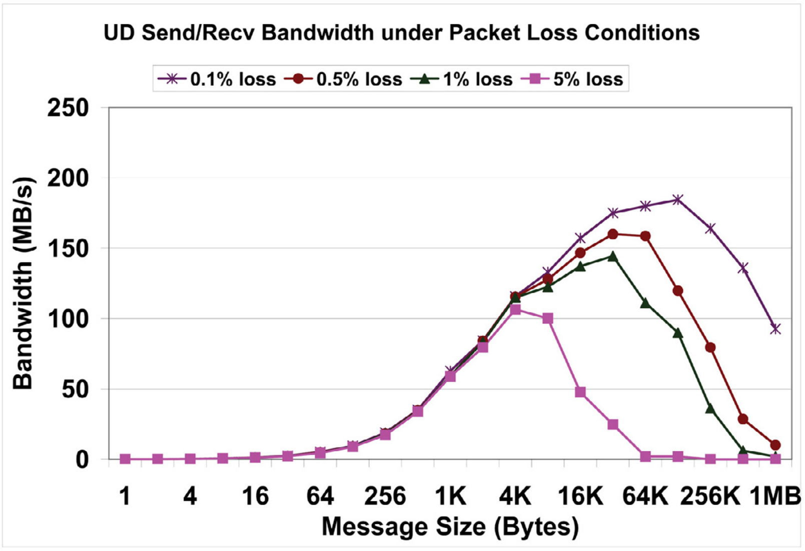

Fig. 8. UD send/recv bandwidth under packet loss conditions.

statistically appropriate manner in order to approximate typical packet loss in real-world situations. The packet loss percentage rates are chosen based on real-world network packet loss records [46]. A loss rate of 0.1% is similar to that of intra-U.S. web traffic, while a 0.5% loss rate is in line with expectations of loss between a western European-U.S. transmission. Larger packet loss rates of 1–5% approximate traffic in Africa and parts of Asia, with African nations typically having rates on the higher ends of the scale.

The bandwidth of UD send/recv datagram-iWARP in Fig. 8 illustrates the observations made in Packet Loss Considerations subsection of the Methodology section, where the loss of a single segment of data of a message causes the loss of the entire message. This loss in turn causes observable losses in throughput for medium-sized messages and catastrophic loss in performance for large messages.

As seen in Fig. 9, a noticeable drop in performance occurs at the 64 kB threshold as messages exceed the maximum-sized MTU of UDP. This situation requires that multiple UDP messages make up a single iWARP-level message. For messages under 64 kB and greater than the network MTU of 1500 bytes, segmentation at the UDP layer must occur. The UDP layer handles all the segmentation and reassembly, resulting in the delivery of a single large message to the upper layer. UDP deals with packet loss of these network MTU-sized packets by dropping the entire UDP layer message. Therefore, packet loss for larger iWARP messages is problematic because there are many opportunities for a completely dropped message if many UDP 64 kB messages need to be

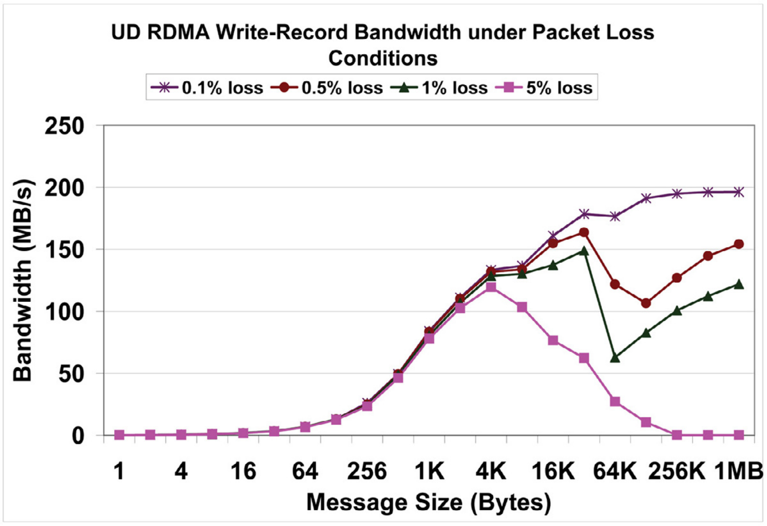

Fig. 9. UD RDMA Write-Record bandwidth under packet loss conditions.

reassembled. Loss of a final UDP-level message is especially unfortunate, resulting in a larger impact on performance than loss of other messages, because the last message indicates to the target node that the operation is complete (it is the last segment); without this packet, the operation must reach a timeout value associated with it before it can deliver the partial message.

The partial-placement feature of RDMA Write-Record shows its advantage in handling packet loss conditions. Segments (up to 64 kB) are placed in memory as they arrive and their reception and location are recorded. Therefore, all the 64 kB portions of a message that arrive without being dropped can be delivered even if they are not sequential. Arguably, high loss rates can still have a significant effect on bandwidth, since some 64 kB UDP messages must arrive without being dropped in order for a partial message to be placed into memory. Nevertheless, the partial-placement method available with RDMA Write-Record provides clear benefits in terms of overall throughput for large messages for networks that experience packet loss. The partial-placement method allows large message sizes to be of use in the most typical packet loss scenarios for Internet traffic in the majority of the world. Without a partial-placement feature large message sizes are impractical for use over lossy networks.

3.3. Datacenter application results

To test the performance of datagram-iWARP with commercial applications, we used VideoLan's VLC, a popular streaming media application [48], and SIPp [11], a framework for load-testing SIP servers. All tests were run using nodes from cluster C1.

3.3.1. Datacenter performance results

In order to assess datagram-iWARP's real-world performance benefits for a streaming media application, VLC's UDP streaming mode was compared with an RC compatible mode (HTTP-based) for a UD vs. RC comparison. Fig. 10 shows the time required to buffer a video stream over VLC using the four modes of operation of iWARP. The UD modes of operation result in a 74.1% reduction in buffering time over the HTTP-based RC alternative. This increase in throughput for the system of almost three times that of the RC method is partially due to the overhead involved in the HTTP-based method; therefore, while UD iWARP shows a performance benefit, it is not entirely responsible for the entire increase. However, this difference represents the approaches commonly used for each transport, therefore, the difference between the performance of the two transports reasonably represents real-world deployment of video streaming.

The performance difference between send/recv and RDMA Write-Record is minimal. This result is due to the need in the software socket interface to provide support for many buffers passed into a single socket. Re-exchanging (advertising) remote buffer locations for every new buffer passed to the interface is expensive. Therefore, in order to reduce overhead, the data is transmitted to a prenegotiated buffer for that socket. For a hardware implementation, this takes the form of a rotating group of prenegotiated buffers for data transfer, and individual application-level sockets are mapped to independent buffer pools. For the software implementation, this results in similar performance to that of send/recv. Therefore, the UD results should be viewed as a single performance number compared with the similar RC numbers for both methods. A hardware solution will allow for a greater differentiation in real-world performance, by using hardware-level zero-copy for large message sizes and hardware buffered copy for smaller messages.

We have also examined SIPp for the base response time for interaction with the server. For a server under light load, the request response averages seen in Fig. 11 were obtained. We observe that the UD-iWARP response time is a 43.19% improvement over that of RC-iWARP for send/recv and 43.24% improvement for RDMA Write (Record).

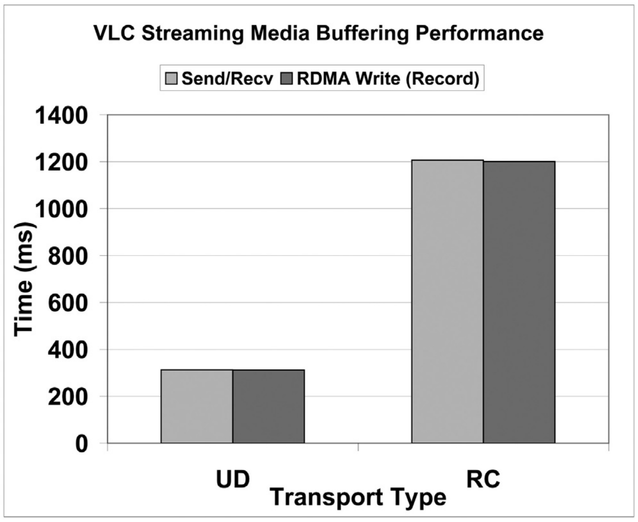

Fig. 10. VLC UD streaming vs. RC-based HTTP streaming.

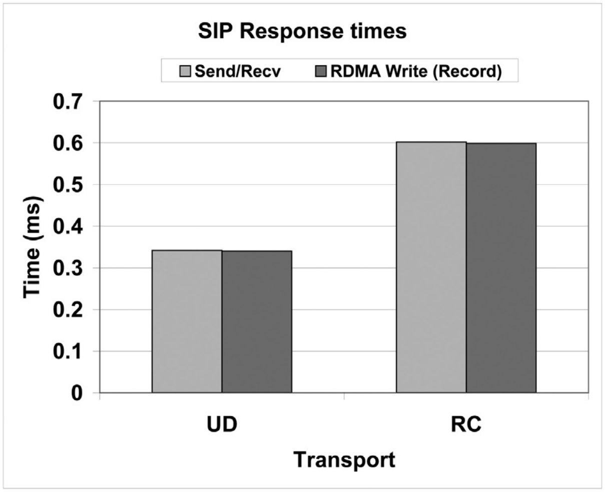

Fig. 11. SIP response times.

3.3.2. SIPp memory usage results

In order to study the memory usage characteristics of SIPp, a server and client were generated and configured by using a basic SipStone client-server test. The memory savings were calculated by using the sum of the SIPp application memory usage and the allocated slab buffer space used to create the sockets. Fig. 12 shows a memory savings of 24.1% at a load of 10,000 clients. Theoretical calculations based solely on the iWARP socket size (using one socket per client) show that such a case should result in a 28.1% memory savings over RC. The 4% difference between these numbers can be attributed to the application's memory usage associated with the datagram sockets. The datagram sockets used by SIPp require additional bookkeeping per socket in order to keep track of the states of the calls over the UDP ports. This amount of bookkeeping causes the larger gaps between the observed memory savings and the theoretically achievable ones with smaller numbers of clients. The overhead of some basic bookkeeping structures reduces the gains at low client levels, but becomes amortized out with larger client counts. The amount of memory used in this test was not excessive, but it can become a significant amount of overall resources on systems supporting hundreds of thousands to millions of clients.

3.3.3. Datacenter application discussion

The datacenter applications tested demonstrate that applications that are data loss tolerant can take great advantage of RDMA Write record and datagram-iWARP send/recv. Combining these observations with those made for the packet loss testing, we can conclude that datagram-iWARP can be an effective transport for WAN datagram-based communications and that it adapts well to the sockets network programming paradigm. As datagram-iWARP demonstrated memory usage savings over traditional iWARP,

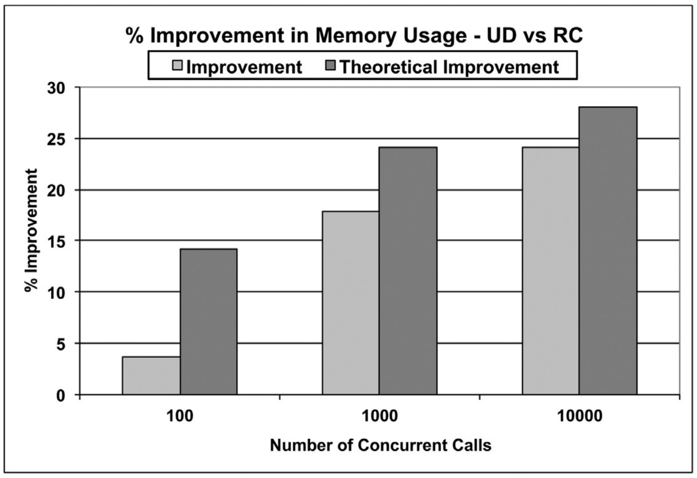

Fig. 12. SIP improvement in memory usage using send/recv datagram-iWARP over traditional iWARP.

it is of interest to explore the scalability of other applications that cannot take full advantage of datagram-iWARP's capabilities because of their requirement for reliable data transmission. We explore such applications next.

3.3.4. MPI-layer microbenchmarks

Since the MPI layer is an important communication middleware, it is of interest to observe the microbenchmark results for latency and bandwidth at this layer. These results are for MPI, which interfaces with the iWARP software implementation through a lightweight OF verbs interface that adds minimal overhead. The majority of the difference between the verbs-level tests and the MPI-level benchmarks is caused by the overhead of the MPI middleware layer itself. We note that MPI is adding some basic lightweight reliability mechanisms that involve some additional overhead as well. These methods have been seen to be effective for the LAN environments that MPI typically runs over. Unfortunately, RMDA Write-Record does not provide as much of a benefit to MPI as it does to datacenter applications, since MPI applications cannot tolerate packet loss. Some mathematical models can tolerate data loss during specific computational phases, and these applications could be adapted to use both reliable and unreliable RMDA in the future. Therefore, much of the benefit in terms of performance for MPI is a direct result of the use of datagrams. This motivates the use of unconnected transports with lightweight reliability and confirms the findings of other researchers when using datagrams over other networks for MPI [27].

The ping-pong latency over MPI is shown in Fig. 13 for send/recv-RDMA Write (Record) modes of iWARP operation. They use the Eager and Rendezvous protocols for small and large messages, respectively. The Eager protocol uses send/recv, while the Rendezvous protocol uses RDMA Write (Record). The threshold (switching) point between the protocols for these tests is a 64 kB message size, since this was shown to be a good operational point for RDMA Write (Record) operations. All results were run with a minimum of 10K iterations over each message size to ensure that the results are accurate representations of network performance. Results show the superiority of the datagram-mode MPI performance over the connection mode, which is carried mainly from the verbs performance benefits, as discussed earlier.

Alternative methods have been developed for RDMA fast-path transmission, which lowers RDMA latency times by using a predetermined set of buffers. Such approaches typically are used for leveraging RDMA for smaller message sizes. We plan to explore adding fast-path-like functionality in future datagram-iWARP implementations.

Fig. 14 shows the bidirectional bandwidth for the UD and RC modes. The bidirectional bandwidth microbenchmark uses two pairs of processes on two nodes, communicating in opposite directions. One of the processes of each pair posts a window of nonblocking receive calls (if applicable), while the other posts a window of nonblocking send (or RDMA) calls that synchronize at the end of the test. MPI in UD mode offers a higher bidirectional bandwidth for all cases compared with that of the RC mode. The improvement for large messages (using RDMA Write-Record) is 20.7%. Lighter protocol processing and minimal reliability measures over a relatively error-free connection are the advantages of UD-based communication that enable the MPI microbenchmark to push more data on the network in each direction.

3.3.5. Microbenchmark discussion

Overall, we observe that datagram iWARP demonstrates superior latency and bandwidth over traditional connection-based iWARP. Since the bottleneck to performance in this software implementation is clearly the operating system communication stack, the implementation of datagram-iWARP in hardware is expected to provide benefits similar to or better than those of the software implementation. The reader is reminded that these numbers correspond to a software implementation on top of the kernel networking stack and therefore are worse than those that would be obtained using a hardware solution.

RDMA Write-Record has a performance benefit at the verbs layer for its potential bandwidth, especially at typical WAN message sizes. We have observed that this benefit is lessened when paired with MPI middleware that requires Rendezvous

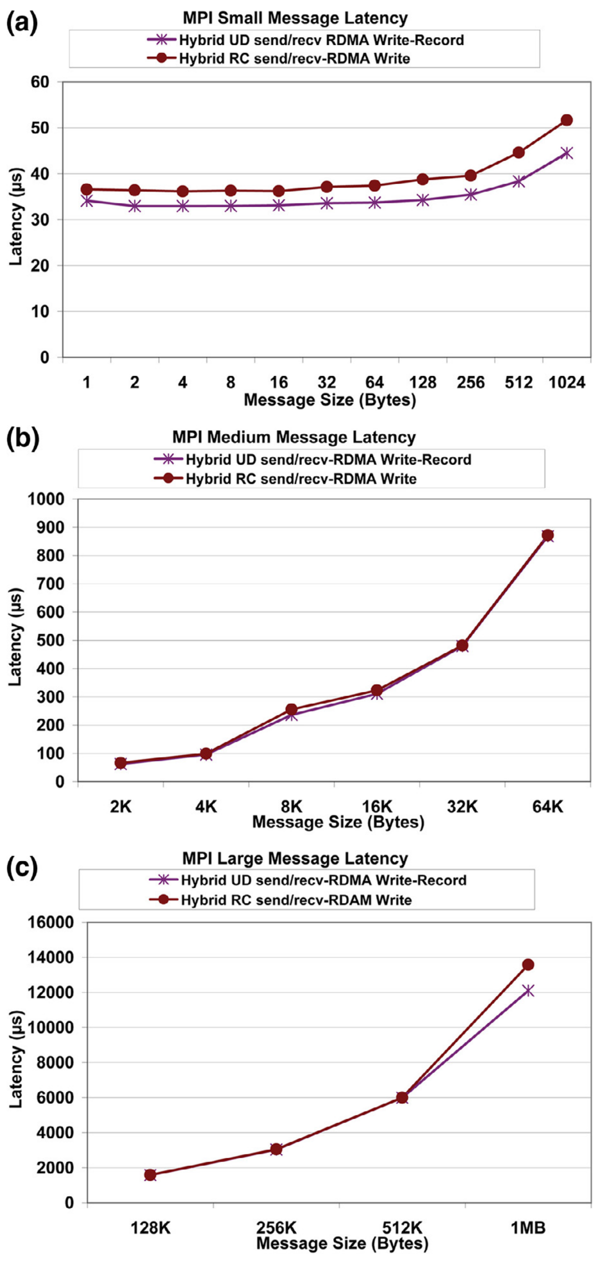

Fig. 13. MPI ping-pong latency.

RDMA traffic. However, approaches such as Eager fast-path RDMA Write operations using pools of buffers could help leverage this performance benefit by reducing the amount of required internode traffic. In Ethernet networking in a commercial environment, the bandwidth performance of RDMA Write-Record can be harnessed and is much more important than the latency performance, since even moderately sized transmission ranges lessen the impact of the lowered latency at the system side. Sustainable high throughput is an important factor in such environments, particularly in the presence of errors.

3.4. MPI application results

Several high-performance scientific computing applications were used for testing over MPI. They were run over a modified MVAPICH-Aptus implementation that was altered to allow for the use of send/recv, as well as the addition of a new transport/protocol path to allow for the use of RDMA Write-Record. While the primary focus is on increased scalability, performance numbers are also presented to demonstrate the potential benefits of using datagrams.

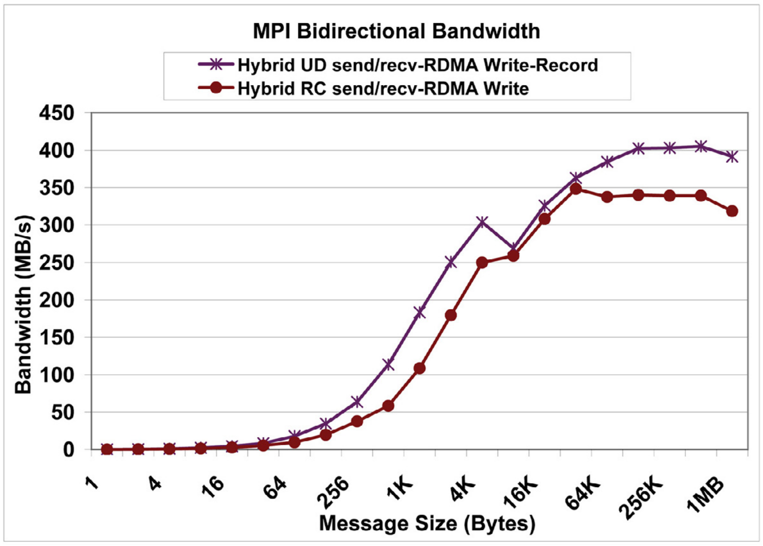

Fig. 14. MPI bidirectional bandwidth.

3.4.1. Application memory usage

The main motivations for datagram-iWARP in an HPC environment are to improve its memory usage in order to make it scalable for large-scale parallel jobs. System-level networking sockets and their associated buffers are handled at the OS level through the slab memory allocation system in Linux. Preallocated pools of socket buffers are registered through the slab, but this preregistration hides the memory footprint of the socket buffer's contribution to overall memory usage at the application level. Only when the slab needs to allocate more space is there a significant change in networking memory usage at the OS level.

Most of the problems with scalability occur at the MPI layer. MVAPICH is designed to provide very high levels of performance and subsequently allocates many resources to communication. For Eager protocols, provisions must be made to allocate buffers for incoming data of various sizes. In order to accommodate such provisions, MVAPICH preallocates a number of general buffer pools with different sizes for each process. When a QP is established, a number of buffers are picked from these pools and preposted as receive buffers to the QP. A default number of 95 receive buffers are picked from the pools and posted to the QP. With an average size of 8 kB for each buffer, approximately 800 kB of memory are required per connection for each process. The reuse of a UD QP for communicating with several other nodes reduces the overall memory allocated to these buffers by not requiring a preposted set of receive buffers for every other node that a process must communicate with.

In order to measure the memory usage for the software-based iWARP, the total number of memory pages allocated to each MPI job in Linux is aggregated. Fig. 15 shows the improvement in application memory usage of send/recv datagram-iWARP over RC iWARP. The results show that as the number of processes increases, so does the percentage of memory saved. These results demonstrate the scalability of datagram-iWARP, where the advantage of having fewer pre-posted buffers per node increases with the total number of nodes in the system.

Applications that do not perform communication between many computational nodes have lower memory savings benefits than do those that are more highly interconnected. This situation occurs for many of the NAS benchmarks because the number of connections per node does not scale quadratically with the number of processes. Other applications such as Radix and SMG2000 use more connections and communicate often; therefore, they show larger benefits from using datagram-based communication.

For the RDMA Write-Record versus RDMA Write methods, there is a negligible memory savings from UD over RC, mainly because the buffers are negotiated when using the Rendezvous method, which does not require preposting many buffers because the buffer sizes and locations are allocated and assigned as needed through a Rendezvous negotiation prior to data transmission. Therefore, for such a buffer management system, examining its memory usage is of limited utility.

3.4.2. MPI application performance

The results in this section are for the class B CG, MG, and LU benchmarks from the NAS Parallel Benchmark (NPB) suite version 2.4 [32], as well as the Radix [45] and SMG2000 [2] applications. All results are presented for 4, 8, 16, and 32 processes. The 64-process results are shown for the C2 cluster for send/recv in order to measure its overall scalability for larger systems. The C2 cluster has fewer cores per node and therefore has a higher potential for more internode communication than does the C1 cluster. The percentage improvement in the application runtimes is shown alongside the percentage improvement in overall communication time. Communication time was measured as the time spent performing all communication primitives: MPI blocking and nonblocking send and receive and MPI wait calls.

The results for UD vs. RC send/recv iWARP MPI application results are shown in Figs. 16 and 17. The results for both communication time and application runtime show that the performance benefits seen from the microbenchmarks for

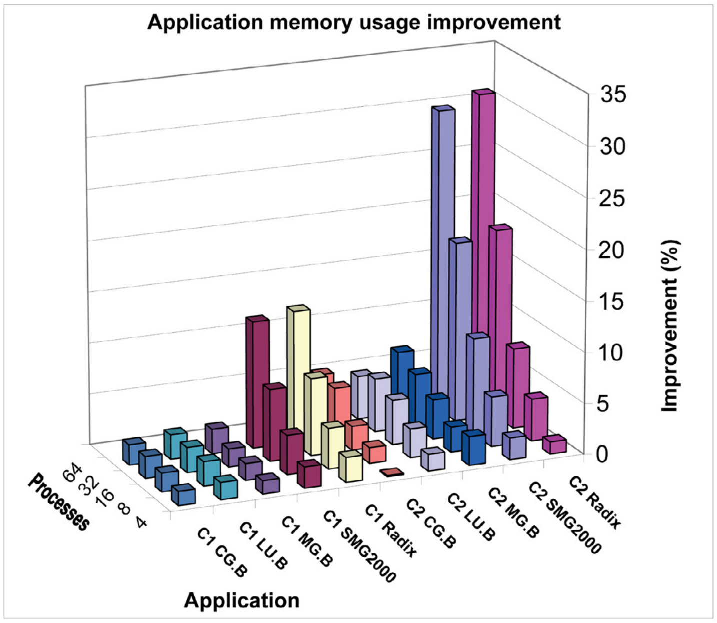

Fig. 15. Memory usage improvement for send/recv.

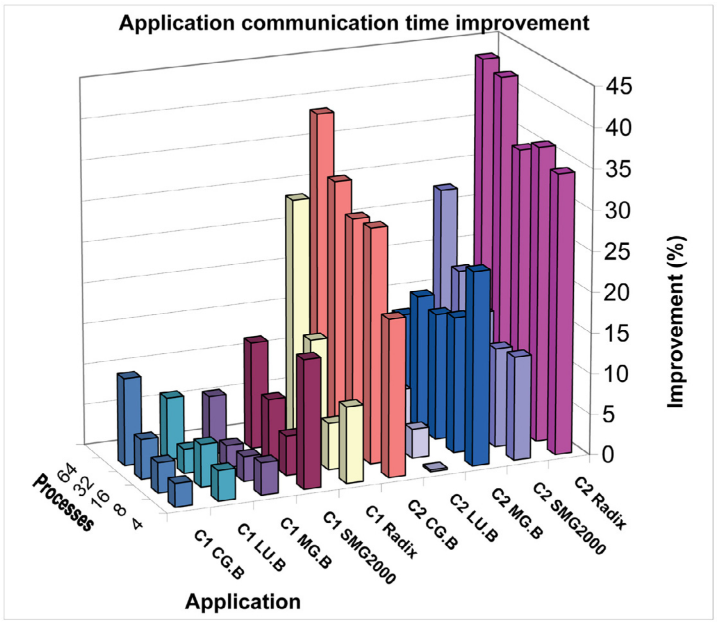

Fig. 16. MPI application communication time improvements for send/recv.

datagram-iWARP have translated into improvement in the application domain. The improvement seen in the C2 cluster is superior to that of the C1 cluster, mainly because of the C2 cluster's increased communication activities.

The improvements possible using RDMA Write-Record over the RC-based RDMA Write were also examined for MPI. This involved setting a threshold for a Rendezvous protocol that uses either RDMA Write or RDMA Write-Record from the iWARP stack. For messages under the threshold size, UD or RC send/recv is used; for those at or above the threshold size, an RDMA Write-Record or RDMA Write operation is used. For this, the same set of applications was run on cluster C1, the only cluster available for testing at the time, with the results presented in Figs. 18 and 19. The Rendezvous threshold was set for a 64 kB message size for those applications that send large messages. This threshold was chosen because it is shown to be an excellent performance cross over point between send/recv and RDMA Write-Record, as well as RC send/recv and RDMA Write in the microbenchmark tests. Some of the applications under test do not send messages of a size greater than 64 kB; therefore, in order

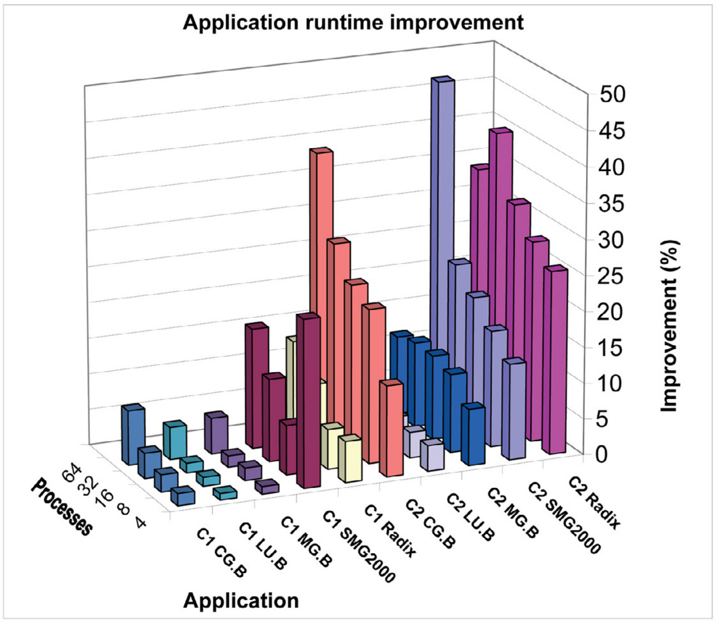

Fig. 17. MPI application runtime improvements for send/recv.

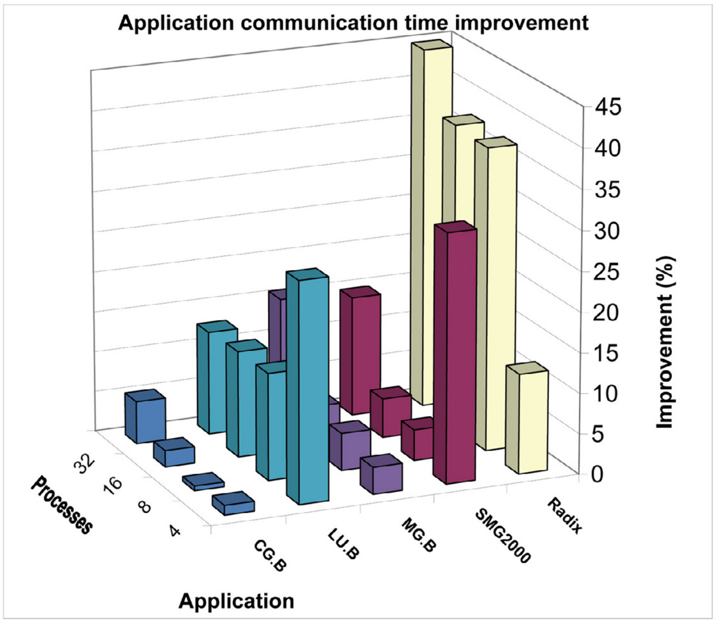

Fig. 18. MPI application communication time improvements for hybrid UD send/recv-RDMA Write-Record on cluster C1.

to assess the performance of RDMA Write (Record), the threshold was not set at 64 kB but instead was lowered to 8 kB for the application tests, since this is the default threshold for the MPI implementation used for testing.

By comparing the percentage improvement of the applications for the pure send/recv solution in Fig. 17 with the hybrid send/recv and RDMA Write-Record method in Fig. 19, we can see that some of the same patterns occur. Overall the improvement of the hybrid UD send/recv-RDMA Write-Record method over that of hybrid RC send/recv-RDMA Write is better than that of pure send/recv UD over RC. We still see excellent percentage improvement in application runtime for 4 processes of SMG2000, like that seen for pure send/recv. The direct performance comparison of runtimes to the send/recv results in Fig. 19 shows that LU, MG, and Radix improvement using the hybrid UD send/recv-RDMA Write-Record method is superior to the improvement of send/recv UD over RC; the other applications show more improvement between the pure send/recv UD/RC modes. The application seeing the greatest benefit from using hybrid UD send/recv-RDMA Write-Record is Radix, which sees a 42.8% improvement in runtime and

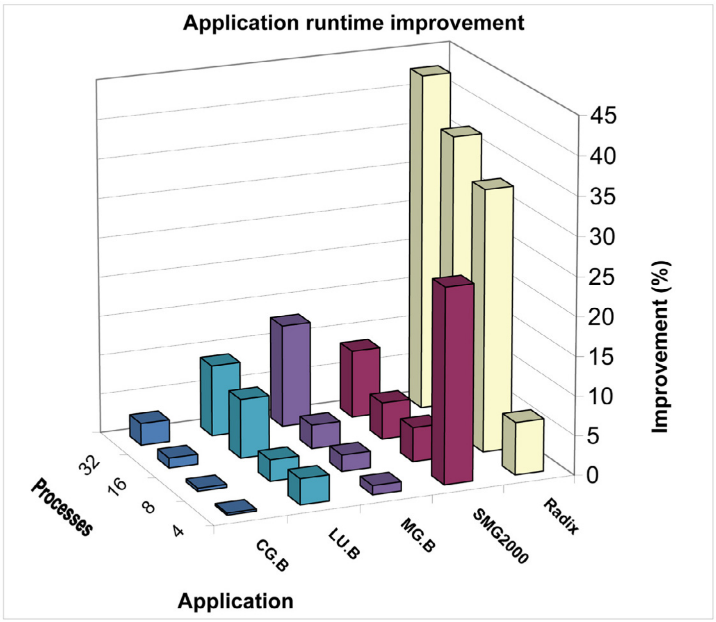

Fig. 19. MPI application runtime improvements for hybrid UD send/recv-RDMA Write-Record on cluster C1.

44.9% improvement in communication time over its RC equivalent mode, while the pure send/recv solution saw only a 14.3% and 29.9% improvement over RC send/recv. The best pure send/recv application, CG, sees 7.6% and 10.6% improvements in runtime and communication over its RC equivalent versus 2.8% and 5.3% for the hybrid case.

4. Conclusions and future work

This paper illustrates the critical design requirements and resulting real-world performance of a proposed next-generation design for RDMA over Ethernet technology. It explores the challenges facing the current iWARP standard and addresses them by extending the standard to the datagram domain. The result is a full-featured datagram-iWARP design. This design includes the first method capable of providing RDMA (one-sided) communication over unreliable datagrams, called RDMA Write-Record. Such functionality can be extended for use on other RDMA-enabled networks.

As a proof of concept, a datagram-iWARP stack was implemented and supplemented with an OF verbs interface for interfacing with ULPs such as MPI and with a socket interface for providing iWARP functionality to existing socket-based applications. An MPI implementation was also modified to work with both send/recv and RDMA Write-Record on top of the OF verbs layer in order to evaluate our implementation in HPC domain.

The datagram-iWARP implementation demonstrates that datagram-iWARP can provide superior performance to traditional iWARP while offering higher scalability. The implementation also includes compatibility with a new set of socket-based applications, as demonstrated by the socket interface layer. The low-level verbs microbenchmarks show a maximum bandwidth improvement of 33.4% and 256% for send/recv and RDMA Write-Record over their traditional iWARP equivalents, respectively. MPI-level benchmarks indicate a possible 14% small-message latency reduction for send/recv, with up to 14.5% and 20.7% large-message bandwidth improvement for send/recv and RDMA Write-Record, respectively. The bandwidth performance of datagram-iWARP when exposed to packet loss was examined. The design alternatives presented for partial data placement in order to help throughput suffering from packet loss provide a good solution for throughput in typical Internet packet loss situations.

Two types of applications were studied: commercial datacenter applications and scientific high-performance computing. For commercial datacenter applications, the iWARP socket interface shows a memory savings of 24.1% and performance improvement of 43.1% for a SIPp server. Streaming media buffering performance shows a 74.1% improvement over RC compared with the relevant operating modes of VLC. Commercial datacenter applications are capable of leveraging all of the benefits of RDMA Write-Record and datagram-iWARP as they can tolerate some level of data loss. HPC applications can benefit from using datagrams with up to a 40% runtime improvement with more than 30% memory savings over connection-based iWARP for some MPI applications using naïve per-process connections on a 64-core cluster for send/recv. Runtime improvements are excellent for RDMA Write-Record over RDMA Write for applications as well. RDMA Write-Record show improved runtime performance even for a smaller cluster. However, the benefits to HPC from using methods such as RDMA Write-Record are constrained by the requirement of reliability. Providing application developers with tools such as RDMA Write-Record will allow for applications that can tolerate data loss to benefit from reduced reliability constraints and allow for greater scalability.

Future work will include the hardware (offloaded) implementation of this protocol for full realization of its capacity, as well as the addition of multicast functionality to the software and hardware iWARP implementations.

RNET's programmable SmartNIC (which we have chosen for our hardware implementation) includes an implementation of iWARP over a reliable UDT protocol (called UDT [14]), which has shown to be beneficial for sustaining high throughput over long-haul WANs (as opposed to TCP), with near zero host CPU utilization. We plan to extend that implementation to support our iWARP over (unreliable) UDP proposal.

In a NIC-offload implementation the following features are expected be superior in terms of performance and scalability to the host implementation:

- At the host side, the existing numerous data copies and OS overheads will be eliminated.
- At the NIC side, multi-stage pipelined and dataflow processing of UDP/IP and RDMA stack by network processing cores will allow for concurrent processing of several RDMA data flows.
- The existing HW accelerators (network processing units) at the NIC side will allow for more efficient protocol processing, data handling and concurrency management.

Moreover, we plan to further analyze the performance of datagram-iWARP on other computing clusters, extending the RDMA Write-Record results by working on additional systems as they become available. Also planned is an extension of the MPI RDMA Write-Record functionality to enable its use of a fastpath-like methodology for decreased latency for small messages.

5. Government license

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada grant #RGPIN/238964-2011; Canada Foundation for Innovation and Ontario Innovation Trust grant #7154; U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under contract DE-AC02-06CH11357; and the National Science Foundation grant #0702182.

References

[1] P. Balaji, W. Feng, S. Bhagvat, D.K. Panda, R. Thakur, W. Gropp, Analyzing the impact of supporting out-of-order communication on in-order performance with iWARP, in: Proceedings of the ACM International Supercomputing Conference (SC07), Reno, Nevada, November 2007.

[2] P.N. Brown, R.D. Falgout, J.E. Jones, Semicoarsening multigrid on distributed memory machines, SIAM J. Sci. Comput. 21 (2000) 1823–1834.

[3] Chelsio Communications Inc, Preliminary Ultra Low Latency Report, 2013, http://www.chelsio.com/wp-content/uploads/2013/10/Ultra-Low-Latency-Report.pdf.

[4] Cisco Systems Inc. Hyperconnectivity and the approaching zettabyte era, May, 2013. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.pdf.

[5] D. Cohen, T. Talpey, A. Kanevsky, U. Cummings, M. Krause, R. Recio, D. Crupnicoff, L. Dickman, P. Grun, Remote direct memory access over the converged enhanced Ethernet fabric: evaluating the options, in: Proceedings of the 17th IEEE symposium on High Performance Interconnects, New York, August 2009.

[6] P. Culley, U. Elzur, R. Recio, S. Baily, J. Carrier, Marker PDU Aligned Framing for TCP Specification (Version 1.0), RDMA Consortium, October 2002.

[7] D. Dalessandro, A. Devulapalli, P. Wyckoff, Design and implementation of the iWARP protocol in software, in: Proceedings of the Conference on Parallel and Distributed Computing and Systems (PDCS'05), Phoenix, AZ, November 2005.

[8] D. Dalessandro, A. Devulapalli, P. Wyckoff, iWARP Protocol Kernel Space Software Implementation, in: Proceedings of the 20 IEEE International Parallel & Distributed Processing Symposium (IPDPS'06), Rhodes, Greece, 2006.

[9] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A. Merritt, E. Gronke, C. Dodd, The virtual interface architecture, Micro 18 (2) (1998) 66–76.

[10] W. Feng, P. Balaji, L.N. Bhuyan, D.K. Panda, Performance characterization of a 10-Gigabit Ethernet TOE, in: Proceedings of the 13th International Symposium on High Performance Interconnects, Stanford, CA, August 2005.

[11] R. Gayraud et al., SIPp Traffic Generator, May 2013. http://sipp.sourceforge.net/.

[12] B. Goglin, Design and implementation of open-MX: high-performance message passing over generic ethernet hardware, in: Proceedings of the 22nd IEEE International Parallel & Distributed Processing Symposium (IPDPS'08), Miami, FL, April 2008.

[13] R.E. Grant, M.J. Rashti, P. Balaji, A. Afsahi, RDMA capable iWARP over datagrams, in: Proceedings of the 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2011), Anchorage, Alaska, USA, May 16-20, 2011.

[14] Y. Gu, R.L. Grossman, UDT: UDP-based data transfer for high-speed wide area networks, Comput. Netw. 51 (7) (2007) 1777–1799.

[15] B. Hauser, iWARP Ethernet: Eliminating Overhead in Data Center Designs, NetEffect Inc., 2006.

[16] S. Hefty, RSockets, Intel Corporation, 2012 https://www.openfabrics.org/resources/document-downloads/presentations/doc_download/495-rsockets.html.

[17] J. Hilland, P. Culley, J. Pinkerton, R. Recio. RDMA Protocol Verbs Specification (version 1.0), RDMA Consortium, October 2002.

[18] G. Huston, TCP performance, Internet Protoc. J. 3 (2) (2000) 2–24.

[19] IEEE standard for local and metropolitan area networks virtual bridged local area networks – amendment: priority-based flow control – 802.1Qbb. http://www.ieee802.org/1/pages/802.1bb.html.

[20] IEEE standard for local and metropolitan area networks—virtual bridged local area networks – amendment: 10: congestion notification – 802.1Qau. http://www.ieee802.org/1/pages/802.1au.html.

[21] IEEE standard for local and metropolitan area networks—virtual bridged local area networks – amendment: enhanced transmission selection – 802.1Qaz. http://www.ieee802.org/1/pages/802.1az.html.

[22] IEEE standard for station and media access control connectivity – 802.1AB. http://www.ieee802.org/1/pages/802.1ab.html.
[23] INCITS – Technical Committee T11. ANSI standard FC-BB-5 – fibre channel over ethernet (FCoE). http://www.t11.org/ftp/t11/pub/fc/bb-5/09-056v5.pdf.
[24] InfiniBand Trade Association, InfiniBand Architecture Specification, Vol. 1, Release 1.2.1, November 2007.
[25] Internet Engineering Task Force, May 2013, http://www.ietf.org.
[26] Internet Engineering Taskforce, Transparent interconnection of lots of links (TRILL). http://www.ietf.org/dyn/wg/charter/trill-charter.html.
[27] M. Koop, T. Jones, D.K. Panda, MVAPICH-Aptus: scalable high-performance multi-transport MPI over InfiniBand, in: Proceedings of the 22 IEEE International Parallel and Distributed Processing Symposium (IPDPS'08), Miami, FL, April 2008.
[28] M. Koop, S. Sur, Q. Gao, D.K. Panda, High performance MPI design using unreliable datagram for ultra-scale InfiniBand clusters, in: Proceedings of the 21 ACM International Conference on Supercomputing (ICS07), Seattle, WA, June 2007.
[29] B. Metzler, P. Frey, A. Trivedi, A software iWARP driver for OpenFabrics, in: presented in OpenFabrics Alliance 2010 Sonoma Workshop, March 2010.
[30] MPI Forum, MPI: A Message Passing Interface standard, v 2.2, September 2009.
[31] S. Narayan, S. Yhi, TCP/UDP network performance analysis of windows operating systems with IPv4 and IPv6, in: Proceedings of the 2 International Conference on Signal Processing Systems (ICSPS), Dalian, China, July 2010 vol. 2, pp.V2-219,V2-222.
[32] NAS Parallel Benchmarks, version 2.4, May 2013, http://www.nas.nasa.gov/Resources/Software/npb.html.
[33] Network-Based Computing Laboratory, MVAPICH: MPI over InfiniBand, iWARP and RDMAoE, Ohio State University: http://mvapich.cse.ohio-state.edu/.
[34] Network Working Group, Stream Control Transmission Protocol (SCTP), edited by R. Stewart, IETF RFC4960, September 2007.
[35] Ohio Supercomputer Center, Software implementation and testing of iWARP Protocol, May 2013, http://www.osc.edu/research/network_file/projects/iwarp/iwarp_main.shtml.
[36] OpenFabrics Alliance, May 2013, http://www.openfabrics.org/.
[37] J. Pinkerton, E. Deleganes, M. Krause, Sockets Direct Protocol for iWARP over TCP, RDMA Consortium (October 2003).
[38] M.J. Rashti, A. Afsahi, 10-Gigabit iWARP ethernet: comparative performance analysis with InfiniBand and Myrinet-10G, in: Proceedings of the 21 IEEE International Parallel and Distributed Processing Symposium (IPDPS'07), Long Beach, CA, 2007.
[39] M.J. Rashti, R.E. Grant, P. Balaji, A. Afsahi, iWARP redefined: scalable connectionless communication over high-speed ethernet, in: Proceedings of the High Performance Computing Conference (HiPC'10), Goa, India, December 2010.
[40] RDMA Consortium, May 2013, http://www.rdmaconsortium.org.
[41] R. Recio, P. Culley, D. Garcia, J. Hilland, An RDMA protocol specification (version 1.0), RDMA Consortium (October 2002).
[42] RNET User-Programmable 10Gpbs Ethernet NIC (SmartNIC). http://www.rnet-tech.com/us/products/2-specialties/2-smartnic.
[43] H. Schulzrinne, S. Casner, R. Frederick, V. Jacbosen, RTP: A Transport Protocol for Real-Time Applications, Audio-Video Transport Working Group, RFC 3550, July 2003.
[44] H. Shah, J. Pinkerton, R. Recio, P. Culley, Direct data placement over reliable transports (version 1.0), RDMA Consortium (October 2002).
[45] H. Shan, J.P. Singh, L. Oliker, R. Biswas, Message passing and shared address space parallelism on an SMP cluster, Parallel Comput. 29 (2) (2003) 167–186.
[46] SLAC National Accelerator Laboratory, PingER, May 2013. http://www-iepm.slac.stanford.edu/pinger/.
[47] H. Subramoni, P. Lai, M. Luo, D.K. Panda, RDMA over Ethernet – a preliminary study, in: Proceedings of the Workshop on High Performance Interconnects for Distributed Computing (HPIDC'09), September 2009.
[48] VideoLan Project, VLC Media Player, May 2013; http://www.videolan.org/vlc/.