High-Performance RDMA using Critical I/O XGE 25Gb Ethernet

Abstract

High performance embedded systems can leverage RDMA (Remote Direct Memory Access) protocols to efficiently move large amounts of data with minimal latency and CPU loading. When combined with ultra-fast transport layers such as 25Gb Ethernet, RDMA can provide unmatched networking performance for embedded processing systems.

High-Performance RDMA using Critical I/O XGE 25Gb Ethernet

Embedded processing systems face an increasing demand to move large amounts of data among sensors and processing elements. Data rates and resolutions are constantly increasing, and these can stress even the best network technologies. And processing elements have become more diverse, including not only traditional CPUs and DSPs, but additional elements such as GPUs and FPGAs.

RDMA (Remote Direct Memory Access) protocols can be leveraged to move large amounts of data with minimal latency and CPU loading. RDMA also supports movement of data directly to/from non-traditional processing elements such as GPUs and FPGAs. And when combined with ultra-fast transport layers such as 25Gb Ethernet, RDMA can provide unmatched networking performance for the embedded system.

Introduction to RDMA and RoCE

RDMA is a long-used standard for the InfiniBand transport layer. When RDMA is instead used over a normal Ethernet network, it is termed RDMA over Commodity Ethernet (RoCE) protocol (aka RDMA over Converged Ethernet). Note that we will use the terms RDMA and RoCE interchangeably, both to indicate the RDMA protocol running over an Ethernet transport layer.

Linux has extensive and mature built-in RoCE support because RoCE is widely used in HPC and large-scale Data Center applications. Critical I/O has implemented a VxWorks RoCE driver that delivers exceptional RoCE performance, leveraging the full RoCE offload available in Critical I/O XGE hardware.

In this paper we discuss RDMA, RoCE, and the associated application to embedded systems. An overview is provided of Critical I/O's RoCE API implementation for VxWorks, as well as the standard Linux RDMA/RoCE implementation. And finally, we discuss RoCE performance, with measurements performed using Critical I/O XGE 25Gb Ethernet hardware. These results show over 3 GByte/s of data transfer performance per port (each direction), with a corresponding CPU utilization of less than 1% for large data transfers, and node-to-node latencies as low as 3.5 usec for small messages.

Introduction to Critical I/O 25Gb XGE Ethernet

Critical I/O's XGE 25GbE XMC is a RoCE enabled Ethernet NIC with two 25GbE ports. The XGE hardware fully offloads RoCE operations, with driver support available for both Linux and VxWorks. Critical I/O's XGE products also support and are fully compliant with existing standard Ethernet protocols and infrastructure. The XGE hardware architecture features a combination of programmable stateless offloads that are natively supported by O/S network stacks, while also offering full RoCE offloads. Critical I/O's software drivers for Linux and VxWorks support concurrent operation of all standard Ethernet operations (TCP, UDP, etc.) with RoCE (as well as UDP Direct, an alternative high-performance protocol described later in this paper).

Figure 1 shows a hypothetical system architecture using 25Gb Ethernet for communication between several subsystems. In this example, the UDP Direct protocol is used for the 25GbE link connecting the Sensor subsystem with the Pre-Processor subsystem, while the low-latency RDMA protocol is used on the 25GbE link connecting the Pre-Processor subsystem with the processor subsystem.

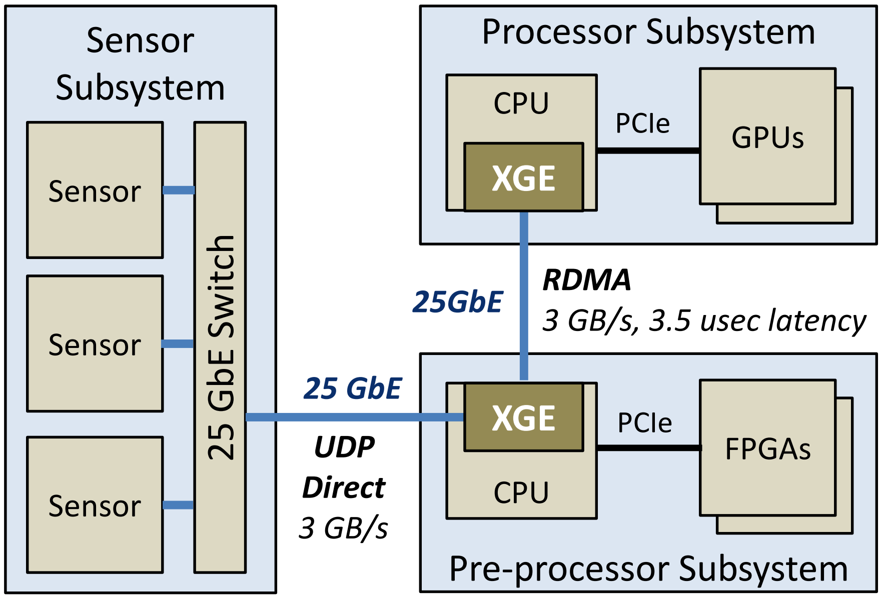


Figure 1. Hypothetical System using 25Gb Ethernet for Subsystem Interconnects

XGE Stateless Offloads for Standard (non-RoCE) Network Traffic

In addition to RoCE offloads, XGE hardware provides protocol offloads for standard networking traffic. For the transmit path, XGE products have hardware assisted checksum computation and Large Segment Offload (LSO) to accelerate transmission of large blocks in data intensive applications. For the receive path XGE products offer hardware assists of Large Receive Offload (LRO) and Flow Steering (FS). XGE LRO aggregates incoming packets into larger buffers in hardware so that the CPU processes a fewer number of buffers, thus reducing a CPU's packet processing overhead. XGE flow steering de-multiplexes packet flows in hardware, thus accelerating traditional receive processing by directly sending a flow to the CPU core that is local to its interrupt, network, and application level processing. Flow steering offload combined with interrupt and application affinity can provide for an improved data rate by lowering the cost of any context switching and reducing cache misses.

RDMA/RoCE Operation

RoCE is designed for efficient data transfers directly between application-level memory buffers. Applications perform RDMA operations such as READ/WRITE that allow direct access to memory locations in each other's address spaces. The RDMA offloads in RoCE enabled NICs translate these operations into hardware DMAs of data directly to/from the desired locations. This combination of low overhead RDMA instructions and offloaded processing of the incoming/outgoing data path provides for highly efficient and low-latency communications.

Because RoCE transfers data directly between application-level memory buffers, it is also possible to send and/or receive data directly from/to buffers that are not located in the host CPUs local memory, for example, memory buffers located with a FPGA or GPU processing node. Traditionally, network data transfers to/from these types of processing elements require first “staging” of data in host CPU memory. With RoCE, host CPU memory staging can be completely bypassed, with data deposited directly where it is needed, overcoming the staging inefficiency of the traditional transfer model.

RoCE supports multiple transport modes such as RDMA_UC (Unreliable Connection), RDMA_RC (Reliable Connection) and RDMA_UD (Unreliable Datagram). The RC transport guarantees that a duplicate message does not arrive at the destination and that messages arrive in order. RC Receivers make use of ACK messages to acknowledge receipt of a message while a NACK is used for a retransmission from the initiator side. Sending of ACK and NACKs is handled by the RDMA hardware offload.

RoCE - Topologies, Flow Control, and Switch Requirements

RoCE can be used between XGE NICs that are connected point-to-point, or in systems using either standard Ethernet switches or enhanced Data Center Ethernet (DCE) switches. DCE switches provide enhanced flow control and congestion management features which are especially useful when there is an expectation of mixed traffic and/or significant network congestion.

In systems where significant network congestion is anticipated, a low loss network is important for best RoCE performance. This means a network flow control method may be required. The default Ethernet flow control method is the 802.3x pause mechanism, which can be used with RoCE nodes which are connected either point-to-point or through Ethernet switches. When an 802.3x pause is asserted it is applied to all traffic on the port, regardless of network traffic type. For improved flow-control granularity, hardware based Priority Flow Control (PFC) and VLANs can be enabled to selectively pause only certain network traffic flows. PFC is especially useful if the link is carrying a significant amount of traditional network traffic in addition to RoCE traffic. And finally, DCE capable switches provide still more features for better flow control and congestion management, including Explicit Congestion Notification (ECN) in addition to PFC.

RoCE – Version 1 vs. Version 2

There are currently two versions of RoCE in use. RoCEv1 is a Layer 2 protocol and is designed for systems where traffic flows within a single subnet. RoCEv2 is a more scalable extension to RoCEv1 that uses a standard IP header, making it a Layer 3 protocol. This means that RoCEv2 traffic can be routed, and thus RoCEv2 can scale to systems spanning multiple subnets. Figure 2 provides an overview of sample interconnects for RoCEv1 and RoCEv2, along with possible flow control methods.

Although more applicable to large IP networks than to embedded systems, XGE also supports Resilient RoCE (RRoCE), a variation of RoCEv2 designed for congested and lossy IP networks. The RoCEv2 specification provides for RoCE Congestion Management (RCM), where the transmission rate is reduced in response to congestion. RRoCE congestion control is NIC hardware based which reduces the latency of the congestion control loop. This allows RRoCE to respond quickly to reduce packet losses while also making use of the fast hardware retransmission logic in the recovery phase. RRoCE requires a switch to support ECN and an Active Queue Management method to detect congestion in switch queues.

RoCEv1 (Layer 2)	**Connection**	**Lossless Link Level Flow Control**					
		Global Pause		**Priority Flow Control (Global Pause OFF)**			
	Point-to-point	ON		PFC on HCAs, VLAN, VLAN Egress Mapping			
	DCE Switch	Host	Switch	**Manual**		**Auto (Remote)**	
		ON	ON	Host	Switch	Host	Switch
				PFC, VLAN. VLAN Egress Mapping (Layer 2 PCP mapping)	PFC, VLAN	VLAN. LLDP, DCBX, VLAN Egress Mapping (Layer 2 PCP mapping)	VLAN, PFC, ETS, LLDP, DCBX
	Standard Switch	ON	ON	N/A		N/A	

RoCEv2 (Layer 3)	**Fabric**	**Connection**	**Global Pause (802.3x)**		**Priority Flow Control (Global Pause OFF)**		**ECN**
	Lossless	Point-to-point	ON		PFC, PFC to Layer 3 DSCP Mapping		optional
		DCE Switch	Host	ON	Host	PFC, PFC to Layer 3 DSCP Mapping, DCBX, LLDP	optional
			Switch	ON	Switch	PFC, LLDP, DCBX	optional
		Std. Switch	Host	ON	N/A		optional
			Switch	ON			optional

Figure 2. RoCEv1 vs RoCEv2 Interconnect Options

RoCE - Implementation and Usage for VxWorks Based Systems

Critical I/O has developed a VxWorks END driver for XGE NICs that allows concurrent operation of RoCE functionality along with standard network functionality (TCP, UDP, etc.). The RoCE API layer implemented for the driver contains a subset of the InfiniBand Verbs API (libibverbs) routines that VxWorks RoCE applications can use for efficient access of the XGE RoCE offload. The API routines exported by the END driver have a 'xe_ibv' prefix in their names but are semantically based on the corresponding libibverbs routines found in the Linux implementation. Figure 3 shows the VxWorks RoCE API routines and their Linux libibverbs counterparts.

RoCE - Implementation and Usage for Linux Based Systems

The Linux XGE RDMA driver permits concurrent operation of traditional networking and RDMA interfaces of the XGE interface. Linux has rich in-built support for RoCE within the Kernel. However, for applications to make use of this kernel support for RDMA, certain user libraries that are a part of the linux-rdma's rdma-core project are also needed. The libibverbs library provides standard IB Verbs (IBV) API (ibv_xxx) while RDMA specific API (rdma_xxx), routines are in the librdmacm library.

API Category	VxWorks RoCE API	Libibverbs/Linux API
Device Access	vctx = xe_ibv_open_device(“xge0”)	vctx = ibv_open_devie(ib_dev_no)
	xe_ibv_close_device(vctx)	ibv_close_device(vctx)
Framework Object Creation	vpd = xe_ibv_alloc_pd(vctx)	vpd = ibv_alloc_pd(vctx)
	vmr = xe_ibv_reg_mr(vpd, …)	vmr = ibv_reg_mr(vpd, …)
	vcq = xe_ibv_create_cq(vctx, …)	vcq = ibv_create_cq(vctx, …)
	vqp = xe_ibv_create_qp(vpd, …)	vqp = ibv_create_qp(vpd, …)
	vmr = xe_ibv_exp_reg_mr(in)	vmr = ibv_exp_reg_mr(in)
Framework Object Modification	xe_ibv_modify_qp(vqp,…)	ibv_modify_qp(vqp,…)
	xe_ibv_req_notify_cq(vcq)	ibv_req_notify_cq(cq)
	xe_ibv_resize_cq(vcq)	ibv_resize_cq(vcq)
Query Framework Objects	xe_ibv_query_qp(vqp,…)	ibv_query_qp(vqp,…)
	xe_ibv_get_port_info(vctx,…)	ibv_get_port_info(vctx,…)
Address Routines	xe_ibv_query_gid(vctx,…)	ibv_query_gid(vctx,…)
	xe_gid_to_wire_gid(gid,…)	gid_to_wire_gid(gid,…)
	xe_wire_gid_to_gid(gid,…)	wire_gid_to_gid(gid,…)
Work Queue Access	xe_ibv_post_recv(vqp,vwr,…)	ibv_post_recv(vqp,vwr,…)
	xe_ibv_post_send(vqp,vwr, …)	ibv_post_send(vqp,vwr, …)
Service	xe_ibv_poll_cq(vcq,…)	ibv_poll_cq(vcq,…)
Framework Object Housekeeping	xe_ibv_destroy_qp(vqp)	ibv_destroy_qp(vqp)
	xe_ibv_destroy_cq(vcq)	ibv_destroy_cq(vcq)
	xe_ibv_dereg_mr(vmr)	ibv_dereg_mr(vmr)
	xe_ibv_dealloc_pd(vpd)	ibv_dealloc_pd(vpd)

Figure 3. VxWorks and Linux RoCE/RDMA APIs

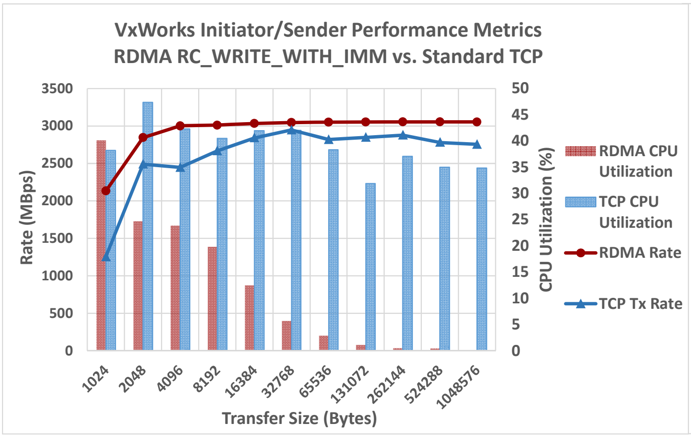
RoCE Performance Measurements for Linux and VxWorks

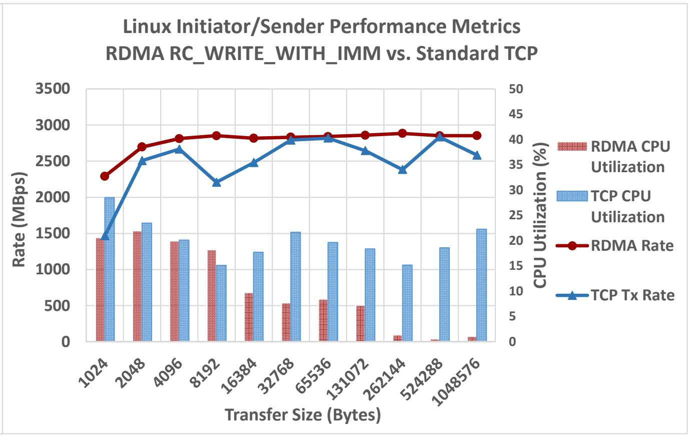
The charts in this section show RoCE performance results obtained using XGE 25Gb Ethernet XMCs, for both Linux and VxWorks. For each OS, three characteristics are measured for a variety of send/receive block sizes: Transfer Rate, CPU Utilization, and Round Trip Latency (transfer time).

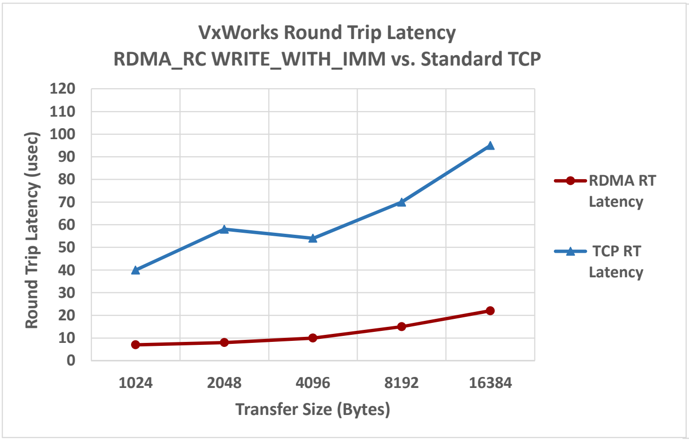
The RoCE results show data transfer rates of over 3 GByte/s with CPU utilization of less than 1% for large block transfers, and message round trip latencies as low as 7 microseconds. The charts below also compare the RoCE performance with the performance obtained with standard TCP socket communications on the same platforms.

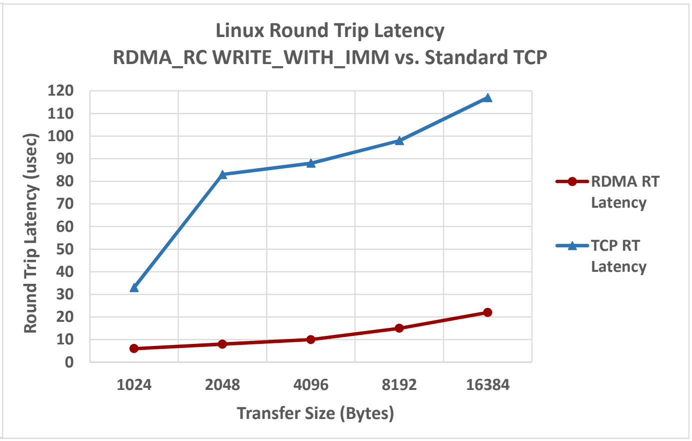
The performance tests are between two nodes, each with an Intel Xeon E3-1225, 4-core processor operating at 3.3 GHz, and a 25GbE XGE interface. The two nodes are connected point-to-point (without a switch). For Linux tests both nodes used Fedora Core 27, while for VxWorks tests the “measured” node used VxWorks 7.0. RoCE on VxWorks used Critical I/O’s RoCE API/driver, while the Linux side used the standard IBV API/driver. All performance tests have used RDMA’s Reliable Connection transport (RDMA_RC). Corresponding Linux TCP performance numbers were obtained using Netperf.

For RoCE Round Trip Latency measurement, we used the RDMA_RC_WRITE_WITH_IMM opcode, so that the latency measurement includes target notification and completion processing delays.









XGE UDP Direct – An Alternative to RDMA/RoCE

Though RoCE based solutions provide outstanding performance, RoCE operation does require RoCE capable NICs at both endpoints of the connection. There are clearly situations where one endpoint may not be RoCE capable. For these situations Critical I/O XGE products support an alternative to RoCE that is based on standard UDP, called UDP Direct.

UDP Direct is a high efficiency networking API/protocol implementation that builds on standard UDP and coexists and operates concurrently alongside other standard O/S networking protocols. By using the UDP Direct API, applications can send/receive large blocks of UDP datagrams by performing DMAs of data directly from/to application memory buffers, completely bypassing all O/S network stack overheads. And, similar to RoCE, these application buffers can be located anywhere in PCIe address space; they are not required to be in the local memory of the CPU that is managing the UDP Direct transfers. This capability can be leveraged in moving data directly in or out of specialized processing elements such as GPUs or FPGAs, with only minimal host CPU involvement.

Summary

The RDMA protocol, running over an Ethernet transport **(**RoCE), delivers high data rates with extremely low CPU overhead by leveraging RoCE hardware offloads, managed through efficient direct access APIs. The XGE 25GbE offers excellent RoCE performance numbers, with send/receive data rates of over 3 GB/s per port, with a CPU utilization of less than 1% for large transfers. As RoCE is widely used in HPC and large-scale Data Center applications, there is extensive and mature built-in RoCE support for Linux. Critical I/O has implemented a VxWorks RoCE driver that delivers exceptional RoCE performance, leveraging the full RoCE offload in Critical I/O XGE hardware. For applications where an endpoint may not be RoCE capable, Critical I/O also provides a UDP Direct API/protocol that allows RoCE-like performance even when interfacing with Ethernet nodes that do not support RoCE.