High Performance Computing with Harness over InfiniBand

A. Valentini[1], C. Di Biagio[2], F. Batino[2], G. Pennella[2], F. Palma[1], C. Engelmann[3]
[1]University of Rome "Sapienza", Italy
[2]Applied Research & Technology Dep. MBDA Italy S.p.a, Italy
[3]Oak Ridge National Laboratory, USA
valentinialex78@virgilio.it, christian.di-biagio@mbda.it, guido.pennella@mbda.it, fabrizio.batino@mbda.it
palma@mail.die.uniroma1.it, engelmannc@ornl.gov

***Abstract*—Harness is an adaptable and plug-in-based middleware framework able to support distributed parallel computing. By now, it is based on the Ethernet protocol which cannot guarantee high performance throughput and real time (determinism) performance. During last years, both, the research and industry environments have developed new network architectures (InfiniBand, Myrinet, iWARP, etc.) to avoid those limits. This paper concerns the integration between Harness and InfiniBand focusing on two solutions: IP over InfiniBand (IPoIB) and Socket Direct Protocol (SDP) technology. They allow the Harness middleware to take advantage of the enhanced features provided by the InfiniBand Architecture.**

I. Introduction

A middleware is defined as *"... a connectivity software that consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network"*. The increasing demand to apply middleware technologies in different application domains, such as real-time systems and embedded systems, has encouraged universities and industry research departments, to develop innovative systems solutions for providing a flexible and extensible Quality-of-Service (QoS) and real-time capability. Generally, newer middleware communication protocols are Ethernet socket based. Even though greater middleware scalability is provided, two performance boundaries are shown: i) TCP/IP protocol introduces a communication overhead and ii) the OS Kernel buffers data before really sending it. Modern network infrastructures, such as InfiniBand (IB) [1], Myrinet [6, 7], Quadrics [9] and iWARP [8] [10], represent a solution to the Ethernet limitations and add other functionality (Remote Direct Memory Access semantic [2, 14], high bandwidth and low switch latency) that is able to increase to network performance. This work, realized with a joint collaboration between the Electronic Engineering Department of the University of Rome "Sapienza", the Applied Research & Technology Department of MBDA Italy S.p.a., and Oak Ridge National Laboratory, present two different solutions to allow the Harness middleware to achieve better performances by using InfiniBand network technology.

A. Modern Middleware: Harness

The Harness project (Heterogeneous Adaptable Reconfigurable Networked Systems), is a result of joint development effort between Oak Ridge National Laboratory, the University of Tennessee, Knoxville, and Emory University. Harness is a distributed, reconfigurable and heterogeneous computing environment that supports dynamically adaptable parallel applications. The main feature of Harness relies on its almost total level of pluggability: the aim is to build a virtual environment that can dynamically change (almost) anything at runtime. In this highly adaptable framework can reside several distributed parallel user applications, all executed over the well known concept of a distributed virtual machine and runtime environment – a PVM inheritance. Harness runs a so called runtime environment (RTE) on every computational unit involved; this environment is the "shell" in which the middleware layer hosts the user applications and the resource management routines that belong to the distributed environment. Every RTE is realized by a Harness kernel which is the core of the unit capable of loading and unloading plug-in modules.

B. Harness at ORNL

Several Harness prototypes have been developed either in Java or in C language. We adopt the C version, which runs on a GNU/Linux-type OS. Specifically, we have chosen the Harness C implementation developed by the Oak Ridge National Laboratory (ORNL) research team. This version is focused on building a lightweight and pluggable middleware layer; it provides a kernel that runs as a Linux daemon process; the kernel manages processes, a thread pool, and it dynamically loads/unloads plug-ins. The process management can fork/execute user applications, it passes arguments, and retrieves output from these external processes. The plug-in loader provides interfaces for loading/unloading modules and for publishing their functionalities to the RTE. The communication facilities are provided by RMIX (Remote Method Interface eXtensible) [11, 12]. RMIX is a dynamic, heterogeneous, reconfigurable communication framework that allows software components to communicate using various RMI/RPC protocols. The RMI (and RPC) paradigm is based on a client-server architecture, where a client invokes a method (or function) at a server-side object. Each client-server pair chooses at compile time or even to negotiates at runtime the most efficient RMI protocol stack that is supported by both sides, while client-side and server-side object stubs remain the same as they only perform an adaptation to the RMIX framework and are not involved in the protocol stack.

C. InfiniBand Architecture

The InfiniBand architecture (IBA) [2, 3] is an InfiniBand Trade Association standard. It is a high-performance network technology designed to realize a connection of processor nodes and I/O devices using a point-to-point switch-based fabric. InfiniBand provides low latency, high bandwidth and Quality of Service (QoS) capabilities. Each element of the InfiniBand System Area Network (SAN) is connected to the network infrastructure by Channel Adapters (CA). In particular, the Host Channel adapter (HCA) is associated with a processing node, while the Target Channel Adapter (TCA) is associated with I/O device. The IBA is built on four layers: Transport, Network, Data Link and Physical layers. These layers are the same of classic OSI stack layers, but in this case, all layers are hardware implemented.

II. Previous Work

Our prior work on a distributed real-time computing environment using Harness [5], focused on making Harness safety critical compliant. As part of this earlier development effort, the modular Harness architecture was preserved and three plug-ins modules were developed to provide:

- a prioritized lightweight execution environment,
- low latency socket communication facilities, and
- local time stamped event logging.

The lightweight execution environment provided by Harness is designed for high real time efficiency based on the thread pool for job processing, which can can control scheduling priority and contention scope. The second plug-in solves the communication problem of the actual RMIX framework (TCP protocol based), which, is not well suited for distributed real-time applications. It introduces a new RMIX provider plug-in (UDP protocol based), while providing a reliable and stream oriented communication. Finally, the last developed plug-in implements a light event logging service based on a temporary shared buffer.

III. Harness Plug-in Implementation

The aim of the presented work is to develop two new plug-ins, which exploit the pluggable nature of Harness in order to implement a set of services enabling its execution across a InfiniBand network. The first solution uses the IP Over InfiniBand technology, which is simple and backward compatible with the original Harness environment. The second solution uses the Socket Direct Protocol (SDP) technology which allows direct communication between the socket API and the InfiniBand (Host Channel Adapter) HCA driver bypassing TCP and IP layers.

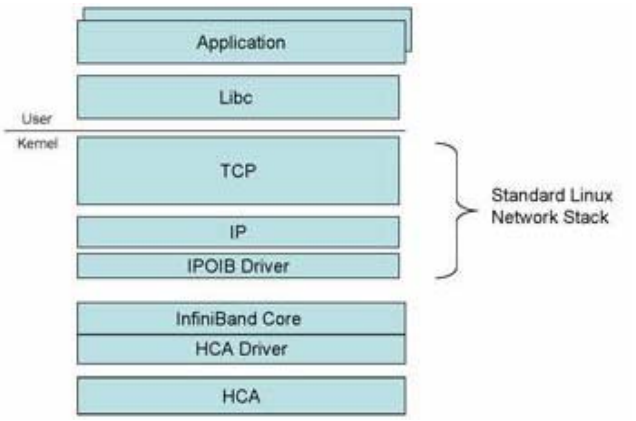

Figure 1. IPoIB Architecture

A. IP over InfiniBand Harness Provider Plug-in

The original Harness RMIX provider plug-in uses TCP over the Ethernet physical layer. Neither Ethernet nor TCP are suitable for high performance deterministic communication. To alleviate this problem we developed a RMIX provider plug-in over an UDP transport layer (see Section 2), but this solution involves the loss of the TCP advances as reliability, order of delivery, etc., characteristics are very important for data sensible applications.

The Harness communication issue can be resolved, with an IB infrastructure that, unlike Ethernet, provides the high levels of reliability, availability, performance.

Our work focuses on developing a new RMIX provider plug-in over the TCP stack, which employs all the TCP protocol functionalities, like reliable and stream oriented communication, but also includes the prioritized lightweight execution environment and the application event logging of our previous Harness version [5].

The IP over IB (IPoIB) solution encapsulates IP packets directly into the IB packets and allows the transport protocol to be used in the communication. In other words, QoS (packet loss, packet order, retransmission, etc.) is guaranteed by the IB layer itself. This encapsulation is operated by the Linux Kernel Module called *IPoIB Driver*, which manages all conventional IP protocol functionalities, such as the Address Resolution Protocol (ARP) and Dynamic Host Configuration Protocol (DHCP). IPoIB solution is fully backward compatible with existing developed Ethernet socket based application. In fact, no changes have to be made to application existing code.

B. Socket Direct Protocol Harness Provider Plug-in

The Socket Direct Protocol (SDP) [4, 14] solution allows socket-based applications to take advantages provided by the InfiniBand Architecture. The sockets direct protocol driver provides a high-performance interface for standard Linux socket applications and provides a boost in performance by bypassing the software TCP/IP stack. The reason that TCP can be bypassed in this model is because InfiniBand hardware provides a reliable transport. SDP works in two different ways: Implicit operation and Explicit operation. The Implicit approach implies to pre-load the *libsdp.so*

library and is able to intercept a TCP socket call within the operating system kernel. The Explicit approach loads dynamically the *libsdp.so* library and foresees a direct SDP invocation, in such a way the traditional socket API parameter AF_INET is substituted by a specific SDP socket definition AF_INET_SDP. Explicit SDP mode involves simple code modifications, but provides a greater application performance. The Explicit operation is used in our test cases.

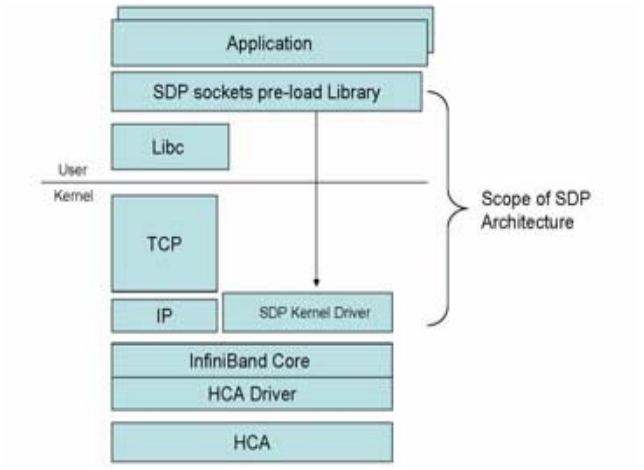

Figure 2. SDP Architecture

C. Enhanced RMIX Structure

Each Remote Procedure Call (RPC) of the RMIX plug-in acts in three phases: 1st: client opening and server connection, 2nd: transmitting call invocation and procedure parameter transmission and 3rd: the client closing. Looking at the SDP approach, it shows complex connection setup operations which introduce high latencies, in this way SDP is not suitable for the RMIX plug-in operation. To alleviate this issue, our work focuses on providing an Enhanced RMIX infrastructure. It provides a new RPC structure, splitting the traditional RPC call in three separate calls: i) client opening and server connection, ii) transmitting call invocation and iii) client close. This enhancement is based on the typical socket operations by first opening the socket, then transmitting all data and at the end closing the socket. Thus, the SDP provider plug-in has been developed in conjunction with the Enhanced RMIX design, the prioritized lightweight execution environment and the application event logging. The Enhanced RMIX design is backward compatible. In fact, no new function calls are introduced, but existing calls are modified. Backward compatibility is very simple to obtain. For example the RMIX *oneway* call now becomes:

oneway(remote_reference, remote_objet_interface, method_to_invoke, array_input_value, input_value_count, client_handle, connection_state)

It can be called in backward compatibility with the parameters:
CLIENT_HANDLE=NULL
CONNECTION_STATE=RMIX_OLD_VERSION

IV. Experimental Results

The development process in industrial, high-performance and time-critical environments implies extensive test activity. We build a test environment that resembles the actual operational environment, both, in hardware and software.

A. Test Environment and Test Case

The Operational Environment of an industrial time-critical application is mainly composed by "Command and Control" (C2) applications. We model a C2 distributed application as constituted by components of one of three types [13]: a *sensor* type component that receives the data from the environment, an *elaborator* component that computes the actions to be taken in response to data received from the sensor, and an *actuator* component that finally executes these actions in order to modify one or more entities of the environment to be controlled. In particular, we develop a test case in order to represent as much as possible an actual industrial C2 application: aircraft guiding. The aircraft guiding test case is built up trough:

- an aircraft, which can turn on the left or on the right and change its velocity,
- a sensor which receives aircraft position and sends command to the elaborator,
- an elaborator which receives data from the sensor and send data to the actuator in order to make the aircraft follow a specific trajectory, and
- the actuator which sends command to the aircraft.

Aircraft, sensor and actuator are installed on a Single Board Computer (SBC) Concurrent VP-347 (Intel PentiumM). The elaborator resides on a different SBC Concurrent VP-347 (Intel PentiumM). Operating system used is CentOS Linux distribution with kernel 2.6.18.

B. Test Results

The goal was to determine the way the packet size and the developed RMIX provider plug-ins impact on the performance of the distributed application in real time environment. Tests show the average round trip time (RTT), reported in micro seconds (μsec), defined as the time that occurs between the data is captured from the *sensor* component and the action taken by the *actuator* component. We test the RTT in the case of i) original Harness 2.0 TCP/IP Provider plug-in, ii) IPoIB Provider plug-in , iii) SDP Provider plug-in based on the new ERP-in. Figure 3 shows the application average RTT latency in an unloaded scenario for three different Harness application packet sizes that are typically used in industrial radar applications. In particular, we are interested in the RTT performance of the RMIX provider plug-in for different packet sizes.

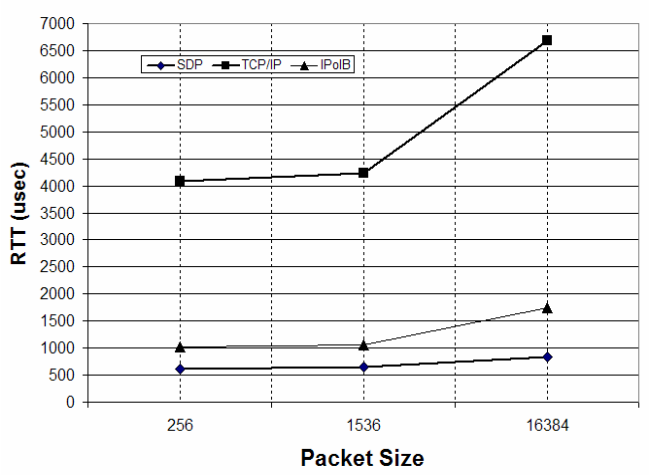

Figure 3. Round Trip Time (µsec)

The TCP/IP plug-in shows higher average latencies than others because i) it does not utilize the prioritized lightweight execution environment, and ii) the communication protocol is TCP over Ethernet. In comparison to the IPoIB and SDP provider, the advantages are evident for any packet sizes. Due to the RPC calling separation, client opening time is mitigated correctly. Note that the SDP provider offers better and more consistent performance. Figure 4 shows the RTT maximum and standard deviation values of the test results, for loaded and unloaded scenarios, which are the most important metrics in real time environments.

		TCP/IP	IPoIB	SDP
Unloaded (µsec)	MAX	34242	1210	1059
	STDEV	7298,95	31	45,6
Loaded (µsec)	MAX	37233	1956	1396
	STDEV	7766,36	152	98

Figure 4. RTT latency (µsec)

It is evident that for the unloaded scenario, the standard deviation values for IPoIB are similar to SDP, while in the loaded scenario the SDP shows better performance. The SDP maximum value is always lower in comparison to the IP maximum values for unloaded and loaded scenarios. This behaviour is due tho the amount of IPoIB driver layers. Finally, it should be noted that the IPoIB and SDP providers show much better performance in comparison to the original TCP/IP provider plug-in, wich also proves our eralier assertion that a provider plug-in build on top of a Ethernet TCP transport layer is not acceptable for distributed real-time applications.

V. Conclusion and Future Work

In this paper, we described an open source runtime communication middleware for distributed real-time application. Within this context, Harness represents an optimal choice because its pluggable architecture offers an dynamic modularity. The Ethernet communication protocol represents a boundary for this middleware, therefore, we have developed a new RMIX provider plug-in that is able to guarantee the integration between Harness and InfiniBand to improve performance for a distributed real-time application (Command & Control) as shown in the test results. The SDP and IPoIB provider plug-ins allow using InfiniBand capabilities without complex source code modification of Harness RMIX, but without taking full advantage of the enhanced features provided by Remote Direct Memory Access (RDMA). Future work may focus on a new Harness RMIX provider plugin that is able to perform RDMA communication for distributed real-time applications. For that, the User Direct Access Programming Library (uDAPL) seems to be the best candidate, as it provides a generic RDMA-capable application programming interface (API). Ongoing related research activities are currently conducted in this direction by MBDA Italia S.p.a Software Applied & Research Dept.

Acknowledgment

The work at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725, was sponsored by the U.S. Department of Energy.

References

[1] IP over InfiniBand Working Group. http://www.ietf.org/html.charters/ipoibcharter.html.
[2] Infiniband Trade Association. http://www.infinibandta.org.
[3] InfiniBand Architecture Specification, Release 1.1, InfiniBand Trade Association, 2002.
[4] Sockets Direct Protocol. http://www.infinibandta.com.
[5] Distributed Real-Time Computing with Harness, Emanuele Di Saverio, Marco Cesati, Christian Di Biagio, Guido Pennella, and Christian Engelmann. Proceedings of the 14th European PVM/MPI Users' Group Meeting.
[6] J. Liu, B. Chandrasekaran, J. Wu, W. Jiang, S. Kini, W. Yu, D. Buntinas, P. Wyckoff, and D. K. Panda. Performance Comparison of MPI Implementations over InfiniBand, Myrinet and Quadrics. In Supercomputing (SC), 2003.
[7] Myricom: http://www.myricom.com/.
[8] RDMA Consortium. iWARP protocol specification. http://www.rdmaconsortium.org/.
[9] J. Beecroft, D. Addison, D. Hewson, M. McLaren, D. Roweth, F. Petrini, and J. Nieplocha. QsNetII: Defining high performance network design. IEEE Micro, 25(4): 34-47, 2005.
[10] Mohammad J. Rashti, Ahmad Afsahi: 10-Gigabit iWARP Ethernet: Comparative Performance Analysis with InfiniBand and Myrinet-10G. Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International 26-30 March 2007 Page(s):1 – 8.
[11] C.Engelmann, C., Geist, G.A.: RMIX: A dynamic, heterogeneous, reconfigurable communication framework. In: Lecture Notes in Computer Science: Proceedings of the International Conference on Computational Science (ICCS) 2006, Part II. Volume 3992., Reading, UK (2006) 573–580.
[12] Engelmann, C., Geist, G.A.: A lightweight kernel for the harness metacomputing framework. In: Proceedings of the 14th Heterogeneous Computing Workshop (HCW) 2005, in conjunction with the 19th International Parallel and Distributed Processing Symposium (IPDPS) 2005, Denver, CO, USA (2005)
[13] Ravindran, B.: Engineering dynamic real-time distributed systems: Architecture, system description language, and middleware. IEEE Transactions on Software Engineering 28 (2002) 30–57
[14] *RDMA Consortium:* http://www.rdmaconsortium.org