A SUCCESS STORY: COLLABORATIVE EFFORT WITH THE INDUSTRY IN ADDRESSING REQUIREMENTS CHALLENGES FOR EARLY ADOPTION OF IWARP IN LINUX

[1]Venkata Jagana, [2]Claudia Salzberg, [1]Renato Recio and [3]Bernard Metzler
[1]*STG, IBM*

[2]*S&D, IBM*

[3]*Zurich Research, IBM*

Keywords: Business process, iWARP, RDMA, OpenSource, OpenFabrics.

Abstract: As customers are embracing Linux in solving business-critical problems, the demand to support innovative and cutting edge technologies is also increasing at a dramatic pace. This has forced the system vendors to offer these technologies much sooner than the traditional cycles allowed. In addition, the open source of different implementations of the same technology by different vendors poses a significant risk in getting an agreement on the common implementation. In order to address the multitude of problems and get an open source implementation of new technology for acceptance into Linux main kernel sooner, we have adopted an innovative method. This method allowed us to work on a common implementation for Linux by avoiding the clash of multiple implementations right from the beginning but of bringing all the relevant vendors, much before the technology gains foothold in the market with any proprietary implementations. In this paper, we'll clearly describe in detail the success story of iWARP support for Linux[1] right from the start of how we have formed a community, generated the requirements that are agreeable to all vendors and open source developers, how it further drove an industry standard to define a programming interface in parallel with the implementation and how the code convergence should happen with an existing Infiniband technology. We'll also describe further how this model can be applied for faster adoption of the upcoming future technologies into Open source-based implementations after addressing the new technology challenges.

1 INTRODUCTION

The Linux operating system has matured from serving simple mail, print, and web services to solving business-critical problems within datacenter environments and thus today has become the mainstream operating system in customer deployments across various industry segments. However, without the active participation and contributions from major vendor companies in the IT industry, Linux wouldn't have matured to the level it is today. In fact, this active involvement from the industry and from end customers was a primary factor in changing how Linux adds newer technologies at a rapid pace within its core operating system. Even though this is a good thing for Linux overall it also brought with it a set of problems, such as causing significant delays in the support of new technologies that requires upstream code support within the core operating system. One of the major causes for this type of delay was due to the conflicts between different implementations of the code submitted by different vendors promoting the same technology. Additionally, these conflicting implementations put Linux at a disadvantage when competing with other competitive OSes within the industry because these other OSes could dictate a specific implementation and thereby enable faster adoption of these leading edge technologies. Obviously, this issue was considered serious and

[1] Linux – Registered Trademark of Linus Torwalds

Jagana V., Salzberg C., Recio R. and Metzler B. (2007).
A SUCCESS STORY: COLLABORATIVE EFFORT WITH THE INDUSTRY IN ADDRESSING REQUIREMENTS CHALLENGES FOR EARLY ADOPTION OF IWARP IN LINUX.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ISAS*, pages 640-645
DOI: 10.5220/0002418806400645

required to be solved with any of the upcoming leading edge technologies for Linux.

An example of leading edge technology, for which multiple conflicting implementations were being pursued in the industry for Linux, was iWARP support using RNICs (RDMA enabled Ethernet adapters). Before describing the solution approach we pursued for this problem, we will briefly introduce the concepts of this technology. We will then describe in subsequent sections the various steps we took to: form a community for developing the code, obtain requirements for the Linux iWARP implementation, define the process for creating the iWARP implementation, execute that process, and ultimately merge the efforts on the iWARP stack with the IB stack. We'll also describe how we ensured the agreed requirements are being carried over in each of these steps.

2 RDMA AND IWARP OVERVIEW

RDMA is a data transfer model that enables out of user space operations to a previously registered user space memory buffer without involvement from either the source or destination's operating system (OS). The RDMA model is dependent on registrations that are stored in memory translation and protection tables (MTPT), which reside in system or adapter memory. These registrations can be used to translate and validate data that is being transmitted from or placed into the registered buffer. The RDMA model provides three basic data transfer mechanisms: Sends, RDMA Writes and RDMA Reads. These data transfer mechanisms are provided to an RDMA-capable adapter through the Send Queue associated with one of many Queue Pairs.

InfiniBand is an example of a protocol stack that supports the RDMA model. Unfortunately, most data center communications use the IP/Ethernet networks stack. Adaptec, Dell, Intel, Broadcom, EMC, Microsoft, Cisco, Hewlett-Packard, NetApp, and IBM founded the RDMA Consortium to create an initial set of specifications for supporting RDMA over existing IP/Ethernet networks. The intention was twofold: to allow vendors to support these initial specifications as is and to seed the IETF's Transport Area's Remote Direct Data Placement Working Group with Internet Drafts that could form the basis for iWARP protocol Request for Comments. The RDMAC specifications are available at the RDMAC Website (http://www.rdmaconsortium.org/home) and the IETF RFCs are available at the IETF Website (http://www.ietf.org/html.charters/rddp-charter.html).

Unlike the IB version of RDMA, iWARP requires integration with the host OS's TCP/IP stack. For example, it requires the handoff of a TCP/IP connection from the host OS to an iWARP QP at the start of an RDMA Stream. An RDMA based stack with defined interfaces to the host OS's TCP/IP stack was required. This stack included defining the required: APIs, interfaces between internal OS components, and interfaces between the OS and the RNIC. The following section will describe the methodology used to create this stack.

3 LAUNCHING AN OPEN SOURCE PROJECT

IBM uses a System I/O Networking team that is responsible for generating the strategic mission and roadmap of IBM servers and storage. The goal of the work is to generate a roadmap that, for the most part, proves to be accurately in line with the unfolding of I/O networking technology improvements and advances. The process used by this team requires input from a diverse set of thought leaders from several disciplines. The input includes: application environment and workload requirements; basic and derivative technology trends; system platform requirements; competitor directions; business model innovations; applicable new and substitute technologies/products; technology/product risk analysis; and the team's capabilities. Additional input is obtained from the larger I/O networking technical community within IBM and outside of IBM. The strategy created by this team is used to drive architecture work, standards work, create plan line items and/or provide input to plan line items. The plan line items are used in the IBM System's Integrated Product Development Process (Grzinich et al., 1997) to drive technologies and products from conception through launch and marker support.

The Systems I/O Networking team discussed the need for an Open Source stack that would support RDMA-enabled NICs (RNICs). A small sub-team was created to define how to enable an Open Source iWARP stack. The team evaluated several alternatives, including having RNIC vendors create their own stacks, leveraging the IB stack, and launching an Open Source project.

In order to avoid the problem of independent implementations of iWARP support for Linux, we decided to launch a community collaborated open source project that would be used as a common platform to gather the requirements and further to implement code in addressing those requirements. Before launching this community collaborated project, we understood that it was critical to gather the requirements from the interested external system and adapter vendors and then have discussions to come up with an agreed list of common requirements that would ultimately become the basis for the launch of the project.

Having laid out that strategy, several system vendors, adapter vendors, and key customers were invited to participate in this collaborative project. Each participating member company was expected to contribute resources to not only discuss and agree upon the requirements, but also help in the subsequent phases to make the overall project a success.

Participating members agreed to pool together the requirements from their respective companies and then came up with a common set of requirements. During this process, members agreed that the process must be flexible enough to change the requirements based on changing needs of the customers and the technology direction. In order to get a better understanding on the approach being discussed, it was important to identify the following key requirements, and some of which will be referenced in later sections.

Requirement R1 – Use Standards defined interface to support the portability aspect

Requirement R2 – RNICPI (RNIC-Programming Interface) is the preferred path for both kernel and user-level APIs. There is no need to provide the user-level APIs direct access (for example, bypassing the kernel) to hardware and Requirement R3 - Agreed to have a thin intermediate user-level access layer to hide the underlying kernel implementation details to avoid the changes in the APIs every time there are changes in kernel code and also to address some inefficiencies through optimizations provided.

Requirement R4 - Agreed to collect all potential major issues (such as TOE service, kernel stack intrusion, ...) internally before going public to discuss with maintainers.

Requirement R5 - OpenRDMA project should aim to support OS version independent of RDMA enablement support, because it would be good for the device drivers porting.

Requirement R6 - Convergence of IB and RDMA interfaces to satisfy the ultimate goal of having a single interface supporting both IB and iWARP. For example, the convergence path needed to be defined for VAPI/IB and RNICPI/iWARP. The expectation was that we could have a single user access layer supporting IB and iWARP interfaces to have single uDAPL or MPI API implementations supporting both types of fabrics.

Having achieved the first and foremost step of arriving at a common set of requirements helped the project to move forward with the architecture definition which was the next big next step before proceeding with the code design and implementation. The architecture for iWARP enablement along with coexistence of Infiniband support was put together by some of the key members of the community and then it was thoroughly reviewed by the broader community of this project to ensure the relevant requirements were taken into consideration. Before proceeding with the next step, we'll briefly discuss the high level design of this architecture to provide a view of the complexity of the requirement dependencies involved in driving this project to success.

4 OPENRDMA ARCHITECTURE OVERVIEW

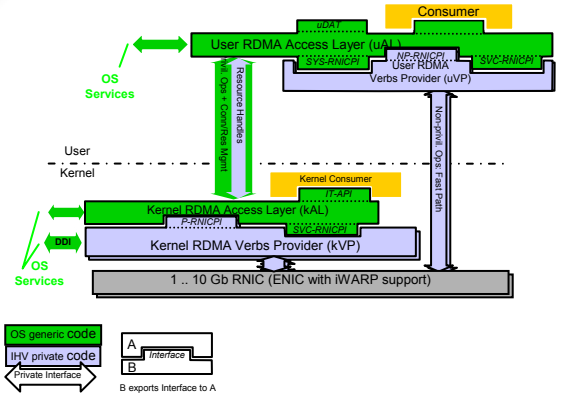

Figure 1: OpenRDMA Architecture.

Fig. 1 summarizes the high level design of the OpenRDMA software architecture. It provides a generic infrastructure (green in Fig. 1) for the integration of IHV (Independent Hardware Vendor) private code (blue in Fig. 1) into the Linux OS. This

infrastructure offers RDMA services for both user level and kernel level RDMA applications also known as RDMA consumers (grey in Fig. 1).

4.1 OS Generic OpenRDMA Components

The IHV independent infrastructure for RDMA support consists of a kernel level Access Layer (kAL) and a user level Access Layer (uAL). On a host system, only one instance of the kAL may exist. The uAL is a loadable library module instantiated by each user level application that is accessing OpenRDMA services. kAL and uAL's are interacting through a generic, but OS-private interface. This interface is further used to support interaction between user-level and kernel-level software components of an IHV.

The implementation of the RDMA API's is part of the OS generic infrastructure. Thus, the kAL may implement DAPL and/or IT-API to serve kernel-level consumers and uAL may implement and expose these services to the non privileged consumer.

4.2 IHV Private OpenRDMA Components

The architecture expects that every RNIC vendor will provide a kernel-level RDMA Verbs Provider module. Optionally, it may provide a user-level RDMA Verbs Provider library for efficient user-level RDMA services. RDMA Verbs Provider modules encapsulate functions that are private to the RNIC implementation, such as QP management and conversion of work requests into WQEs.

The kVP module provides vendor-specific software implementing the semantics of the RDMA verbs as defined in (Jeff et al., 2003). It translates the verb calls issued by the kAL into appropriate actions such as the creation and management of QPs and the insertion and removal of WQEs. All QP, CQ, and SRQ data structures are under direct control of the module and are not accessible to the consumer. While the module's interfaces to the RNIC hardware are IHV-private, its upper interface to the kAL will implement the RNICPI (RNICPI, 2005) as defined by the Open Group.

The uVP contains vendor-specific software for the provision of user-accessible RDMA Verbs services to the uAL and for direct access to the RNIC hardware. It establishes a fast path to the RNIC hardware for all performance critical operations that can be performed without holding exclusive access permissions. This includes all send-and-receive type operations. As with the Kernel RDMA Verbs Provider, all QP, CQ and SRQ data structures are under direct control of the library and are not accessible to the consumer.

Even though the architecture focus has been primarily around Linux, it was easily adaptable to other OS variants for meeting adapter vendor needs.

5 PARTICIPATION IN RNICPI STANDARD EFFORT

There have been several assumptions made within the architecture about the use of either existing or new internal interfaces within the OS kernel and also at the user space. One of the critical internal interfaces required for supporting RNIC verbs is a standard-based interface that would ensure the adapter vendors could easily port their driver across OSes. Since the standardization of this interface required broader industry support, for example, through a standard-based organization such as OpenGroup, a separate effort was needed to be driven outside of this community.

Interestingly enough, at around the time that the OpenRDMA project was launched, there was also an effort brewing within the industry to create a workgroup under OpenGroup to standardize the interface for Unix systems. The objective of this workgroup was to provide an abstract interface based on RNIC verbs and coincidentally, this effort was in line with OpenRDMA community requirement R1. Some of the key community members from this project had started working with OpenGroup in not only providing the requirements of this interface but also contributed to the evolution of this interface definition. Another significant impact that this OpenRDMA community made in the interface definition was to convince the OpenGroup to allow, which is not its normal practice, its draft versions of the standard to be reviewed by this OpenRDMA community to provide its input early in the process before it becomes a standard. This was made possible simply because OpenGroup understood that this OpenRDMA community could develop an early prototype that would validate the interface but also this community established credibility with OpenGroup right from the start of forming this WG through requirements input.

Because there was a good representation of OpenRDMA community members participating in the standard effort of this interface, it made it simpler to get its requirements met in terms of what the broader community would like to see in the interface function calls and the related required arguments. This type of participation and collaborative effort further proves that there can be significant advantages in a collaborative model over traditional development models.

Based on an early definition of RNICPI interface available to the OpenRDMA community, the developers started to contribute the code for components defined in the agreed architecture. In fact, a separate code maintainer for the project was identified and made responsible for ensuring the contributions not only met the required opensource guidelines but also ensured the conformity for agreed architectural definition such that the original requirements were continuing to be met.

6 MERGE OF OPENIB AND OPENRDMA

The project was making good progress but in order to meet the requirement R6 of converging Infiniband (OpenIB) and OpenRDMA for a single common stack, the strategy was to merge these two projects before these independent stacks had fully evolved. However, the challenge, which was recognized as a major issue later, was to find a common ground in merging the different interface APIs adopted for both of these implementations. Since RNICPI was evolving to support both IB and RNIC verbs, but the OpenIB interface API was supporting only IB verbs at that time and the interface APIs were different, it was perceived that the possible convergence approach might be to support through RNICPI rather than modifying the existing OpenIB verbs. However, it was realized that this could be a challenge in getting this approach being accepted upstream as the OpenIB API was already made upstream.

Since the OpenRDMA implementation is completely based upon RNICPI supporting both transport-independent and transport-dependent functions, it was critical to have both transport modes supported in the merged interface. Hence, this approach was absolutely taken into consideration during the merge effort by developing a separate connection manager for each of the underlying transport of Infiniband and iWARP and a generic RDMA connection manager. Even though the Linux implementation ultimately did not pick up the RNICPI primarily because changing to a new interface in supporting the existing IB drivers and upper layer protocols meant more changes everywhere in the stack which is not realistic, it did pick up some of the good concepts from RNICPI.

As the progress of this OpenRDMA project reached a good pace, the OpenIB community was also making good progress in its code acceptance upstream. Although this was good news, it inherently put pressure on OpenRDMA to merge both of the projects sooner rather than later. This merge was required sooner to avoid the discreet implementations of two independent stacks, one for Infiniband and the other for iWARP, to exist in Linux and also to avoid the duplication of the similar components of two stacks.

In order to have a successful merge of these two emerging stacks into a single common stack, the need was to bring in the key community members of both communities onto common ground and quickly agree upon the requirements in a collaborative manner. The leaders of both communities had come to an absolute understanding that this convergence effort must be clearly driven with urgent timelines to avoid further confusion of overlapping projects in the communities.

This agreement has further allowed for quick progress on the convergence aspects from an architecture, interface and implementation standpoint with the better commitments from both communities. This was only possible after ensuring the corresponding requirements were being discussed first and agreed upon by the combined collaborative community. For example, to be specific on the set of requirements agreed upon, one of the few requirements (a) is to combine the best features of both stacks to generate a unified stack that would allow applications and Upper Layer protocols to use a single set of interfaces rather than using a different set of interfaces for Linux. Another requirement (b) is to ensure the existing IB stack functionality is not impacted while the code convergence happens from both of the projects.

The key here was to ensure the set requirements were taken into consideration during the actual code merge process. After agreeing on how to satisfy these requirements, it paved the clear path for the projects to merge, forming the OpenFabrics community (OpenFabrics Alliance). Through the OpenFabrics community, the infrastructure code for

supporting iWARP in an incremental fashion is currently made into Linux mainline kernel.

7 SUMMARY

In summary, the success of this project really proved that the collaboration effort among the vendors and customers through an early participation in an open source project for addressing the requirements meeting most of the needs would really work. But also, this success has set a new trend for vendors who could use the resources much more efficiently without having to duplicate the software effort to enable new hardware and thus avoiding the migration from their own proprietary stacks to an open source stack and this further reduces the burden on the customers in their environments. In the future, this requirement model could easily be applied to other emerging technologies.

ACKNOWLEDGEMENTS

We would like to thank the following folks for their contributions to this effort: Kanoj Sarcar, Krishna Kumar, Pradipta Kumar, Leonid Grossman, Aaron C Brown., Caitlin Bestler, Michael Krause, Thomas Talpey, Tom Tucker and Steve Wise.

LEGAL STATEMENT

REFERENCES

Grzinich, J. C.; Thompson, J H.; Sentovich, M.F.; “Implementation of an Integrated Product Development Process for Systems”, PICMET '97: pp: 427 -430,1997

Jeff, H; Paul, C; Jim, P; Renato, R; “RDMA Protocol Verbs Specification (Version 1.0)”, Apr, 2003

“RNIC Programming Interface (RNICPI) Version 1.0”, OpenGroup, Sep, 2005

“OpenFabrics Alliance”, www.openfabrics.org