SEVENTH FRAMEWORK PROGRAMME
Research Infrastructures

FP7-ICT-2011-7

DEEP

Dynamical Exascale Entry Platform

Grant Agreement Number: 287530

D4.1
Cluster Booster low-level protocol

Approved

Version: 2.0
Author(s): N. Eicker (JUELICH), A. Galonska (JUELICH)
Date: 12.06.2012

Project and Deliverable Information Sheet

DEEP Project	**Project Ref. №:** 287530	
	Project Title: Dynamical Exascale Entry Platform	
	Project Web Site: http://www.deep-project.eu	
	Deliverable ID: D4.1	
	Deliverable Nature: Report	
	Deliverable Level: PU	**Contractual Date of Delivery:** 31 / May / 2012
		Actual Date of Delivery: 29 / May / 2012
	EC Project Officer: Leonardo Flores	

* - The dissemination level are indicated as follows: **PU** – Public, **PP** – Restricted to other participants (including the Commission Services), **RE** – Restricted to a group specified by the consortium (including the Commission Services). **CO** – Confidential, only for members of the consortium (including the Commission Services).

Document Control Sheet

Document	**Title:** Cluster Booster low-level protocol	
	ID: D4.1	
	Version: 2.02.0	**Status:** Approved
	Available at: http://www.deep-project.eu	
	Software Tool: Microsoft Word	
	File(s): DEEP_D4 1_Cluster_Booster_Protocol_v2.0_ECapproved	
Authorship	**Written by:**	N. Eicker (JUELICH), A. Galonska (JUELICH)
	Contributors:	M. Nüssle (UniHD), B. Tweddell (JUELICH)
	Reviewed by:	E.Suarez (JUELICH), F.Schuermann (EPFL)
	Approved by:	BoP/PMT

Document Status Sheet

Version	Date	Status	Comments
1.0	29/May/2012	Final	Submitted to EC
2.0	12/June/2012	Approved	Approved by EC

Document Keywords

Keywords:	DEEP, HPC, Exascale, EXTOLL, InfiniBand, PCI Express

Table of Contents

List of Figures

Executive Summary

The two parts of the DEEP System – Cluster and Booster – will have their own interconnect optimised for the requirements of the specific application kernels, i.e. InfiniBand and EXTOLL respectively. A network bridge between both parts, the Booster Interface (BI), enables all Cluster Nodes (CN) to make use of whole partitions of Booster Nodes (BN) in order to offload highly scalable kernels to the Booster Node Cards.

A key aspect in the design of the proposed network architecture will be a high throughput network protocol allowing to offload the highly scalable application kernels with low overhead from Cluster to Booster. For that, the Booster Interface Card (BIC) is equipped with two PCIe ×16 slots attached to a PLX PCIe switch to bridge both networks via PCIe peer-to-peer communication. To avoid unnecessary utilization of the BIC's processor, data transfers will be based on a combination of Remote Memory Access (RDMA) techniques provided by both networks. As a fallback solution a message-based data transfer solution is also foreseen.

The communication between Cluster and Booster will be divided into two channels: a control and a data channel. The control channel is used to set-up data channels based on RDMA between a CN and a BN. It can also be used as a fallback solution providing a message-based data transfer. For this purpose a BI daemon software will be implemented. Together with appropriate client software running on the CNs and BNs, it will orchestrate the Cluster Booster protocol.

This document describes the requirements, the design and the implementation of the Cluster Booster low-level protocol as it is foreseen for DEEP. Beginning with a list of approaches on implementing the data path from the Cluster to the Booster, the design of a BI daemon software is explained, which enables nodes of both networks to talk with each other. The focus of this task lies on communication initiated from a CN.

This report at hand is intended for developers of high-level communication functionality in the context of the DEEP project. Members of task 4.4 developing the Booster off-load approach as well as the programmers of the global MPI environment from task 4.2 and WP5 shall take this document as a basis for their own work to get an idea of the underlying functionality and how to integrate it in their own software stack.

The DoW foresees that the implementation of the low-level Cluster → Booster protocol is finished in M6. Due to a delay in the procurement and setup of the BIC evaluator this task is about 2 month behind schedule. Instead, the design of Booster initiated communication was advanced. With the availability of the BIC evaluator by end of May the implementation of the bidirectional low level protocol immediately started.

1 Introduction

The DEEP project is intended to develop a novel supercomputer architecture different from the approaches which exist today. While traditional Cluster Nodes can be equipped with accelerators to off-load suitable application kernels, our approach foresees a separate cluster of accelerators called Booster, which can be utilized by all of the Cluster Nodes (CN). Both clusters have different interconnects requiring a bridge, the Booster Interface (BI), for communication between both networks. The data transfer between both fabrics will be realized by the Cluster Booster low-level protocol.

The development of the Cluster Booster low-level protocol is a key part for obtaining a well performing DEEP System. As this software targets the Booster Interface, which has to bridge between the Cluster and the Booster networks, the aim of this development is gaining high transfer rates with low overhead.

In the Cluster part of the DEEP System a switched InfiniBand fabric [1] with fat-tree topology will be used allowing for complex communication patterns. The Booster interconnect will be based on EXTOLL technology [2] implementing a highly scalable 3D torus topology. These two networks come together in the Booster Interface, where an InfiniBand HCA and an EXTOLL NIC are connected via PCIe ×16 links to a PLX PCIe switch. In the final system 8 Booster Node Cards (BNC) housing 2 Intel Knights Corner (KNC) processors each are attached to every Booster Interface. The BI in DEEP will come in form of a so called Booster Interface Card (BIC) set up from minimal hardware required to establish a connection between both networks via PCIe. Due to InfiniBands fat-tree topology all BICs are accessible from all CNs without significant overhead.

For development and testing purposes the BIC evaluator is set-up since final hardware for the BIC is not available before M20. This system consists of three standard x86 servers, each representing a CN, a BN and a BI, respectively. While the CN-machine is equipped with an InfiniBand HCA, the BN hosts an EXTOLL NIC and the BI has both types of network available. Although the BN in the BIC evaluator is not equipped with a processor based on MIC technology, it is expected that the software stack developed and tested on the BIC evaluator can easily be ported to the final architecture.

To enable communication between Cluster and Booster, an interface has to be developed to transparently transfer data in both directions. Technically, this interface has to bridge both networks creating data paths from InfiniBand to EXTOLL.

Our software stack utilizes ParTec's pscom library that is part of ParaStation [3] to create message-based client-server control communication on our BIC-evaluator between the BI, acting as a server, and the two other systems as clients. The control connection is then used to exchange data required for establishing a high speed RDMA/SMFU connection between a Cluster and a Booster Node.

Beginning with a description of the development vehicle this document explores the possibilities on how to transfer data via an InfiniBand fabric into a BNs memory employing the EXTOLL network using PCIe and remote memory techniques. Problems are identified and pre-requisites are defined to extract criteria for the hardware of the final BICs. The design of a BIC daemon software is sketched, in order to illustrate the functionality of high bandwidth and low overhead data transfers from CNs to BNs via the BI.

The prototype system and the software stack mentioned here will be used to evaluate the requirements for the Booster Interface Cards that shall be used in the final DEEP System. On the hardware side these cards have to house two PCIe ×16 slots with an appropriate PCIe root

complex fitting to the requirements specified in section 3.2.1. A processor and main memory is also necessary to run a minimal Linux system and the service-daemon software.

2 Development Environment – The BIC-evaluator

The design and development vehicle used within this task consists of three MEGWARE MiriQuid server systems equipped with standard x86 CPUs. This so called BIC-evaluator will be used to carry out all implementations and tests necessary to fulfill task 4.2. Its hardware was chosen due to the need of a testing environment and the lack of existing Booster Interface Cards or Booster Node Cards equipped with Intel MICs processors. Both are currently under design by Task 3.2 in WP3.

The three machines of the BIC evaluator are representing a CN, a BN and the BI, respectively, as it is foreseen in the final DEEP System. It is sketched in Figure 1.

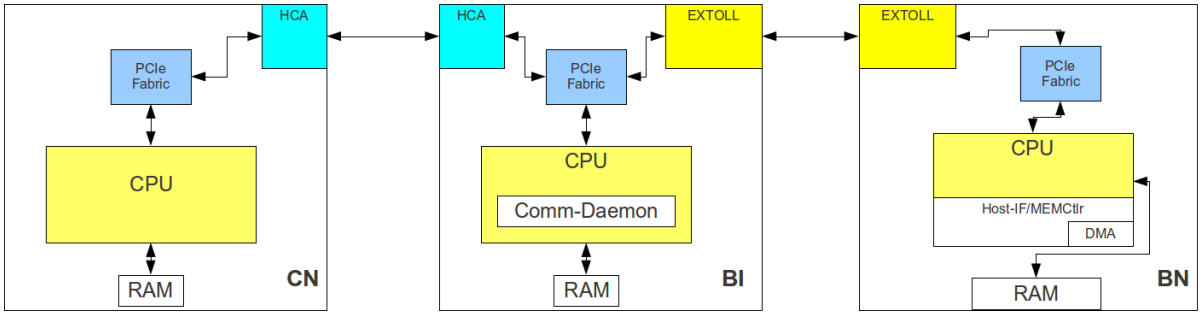

Figure 1: Architecture of the BIC evaluator system

The server representing the CN is equipped with only one InfiniBand HCA, connected to the server representing the BI. The BI server houses an InfiniBand HCA and an EXTOLL NIC, both connected via a PLX PCIe switch. This switch bridges both networks physically. It is controlled by a particular software running as a service-daemon on the BI. The functionality of this software is described below. Finally the EXTOLL NIC is connected to its counterpart on the third server representing the BN.

A report on design decisions taken, the procurement, and setup of the BIC evaluator will be available in Deliverable 6.2 of WP6 later.

3 Cluster Booster low-level protocol approaches

The BIC evaluator will be used to implement and evaluate efficient ways to communicate from CNs connected to InfiniBand with BNs connected to EXTOLL. Within this task, all communication is assumed to be initiated by the CN. In a later document this constraint will be relaxed and even communication initiated by the BNs is taken into account.

To better understand the possibilities, a short overview of the communication features of EXTOLL is given (more details can be found in [2]). Here InfiniBand will be seen as a simple RDMA mechanism between CNs and the BI address-space due to the limited measures to influence the behaviour of InfiniBand HCA. The latter results from the fact that the corresponding firmware is proprietary and cannot be adapted to the requirements of the Project.

There are three standard communication engines present in an EXTOLL device:

- VELO offers send/receive functionality from a sending CPU to receiving main-memory queues, especially for small and medium message sizes.

- RMA offers one-sided put/get operations including a notification mechanism, where the completion of an operation at the requesting, responding or completing host can be signaled.
- SMFU [4] offers a global, non-cache-coherent shared memory. Read or write transactions from the local system can be forwarded to remote nodes. The destination node is selected by a configurable part of the physical address of the transaction.

Within the context of this document, SMFU provides the most promising functionality since it allows to forward DMA read and write transactions of remote devices even to another host, if the PCIe root complex supports peer to peer read and write operations. With that, InfiniBand RDMA techniques on the Cluster side can be used to access memory inside the BN exported into the address-space of the BI.

From this introduction, several possibilities for IB-to-EXTOLL protocols can be derived:

- SW Bridged: The BI hosts a service process, which actively forwards all messages received from a CN to a target BN.
- Leveraging SMFU and InfiniBand RDMA: This strategy will enable a CN's InfiniBand HCA to write/read directly to/from the memory of an attached Booster Node. The SMFU address space maps the BN's memory into the BI's local address-space, allowing for the direct access to this memory from the InfiniBand fabric. The communication is initiated by the CN.
- Remote Operation of InfiniBand RDMA via SMFU: The BN triggers a data transfer on the CN by accessing its HCA registers via SMFU.
- Hybrid Approaches: Here the first two techniques are combined and the right strategy is chosen based on message size and latency. If a RDMA/SMFU transfer would be unnecessary i.e. if the message size is smaller than a pre-defined threshold, the SW bridged solution is chosen and the message will be forwarded by the BI.

3.1 SW Bridged Cluster-Booster Communication

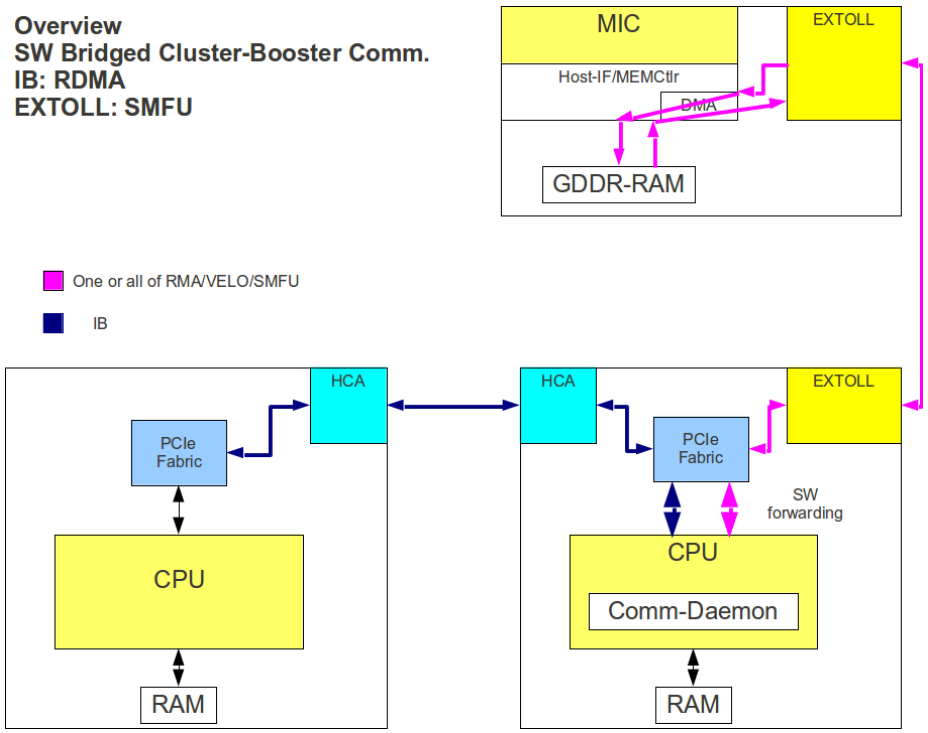

Figure 2: SW bridging concept

The most straight forward version from an implementation point of view is to support communication via SW bridging (Figure 2): Here, InfiniBand transfer operations are terminated in the BI. A service daemon residing on the BI then actively re-sends the data via the EXTOLL network to its final destination on the BN (Figure 3) for write operation or the CN (Figure 4) in case of a read operation. Although this option enables all necessary functionality it probably increases latency and lowers bandwidth. It is still worthwhile to mention it here, as it provides a low-risk fall-back solution. Some evaluation will also be done to assess the gains associated with better solutions (see the following subsections).

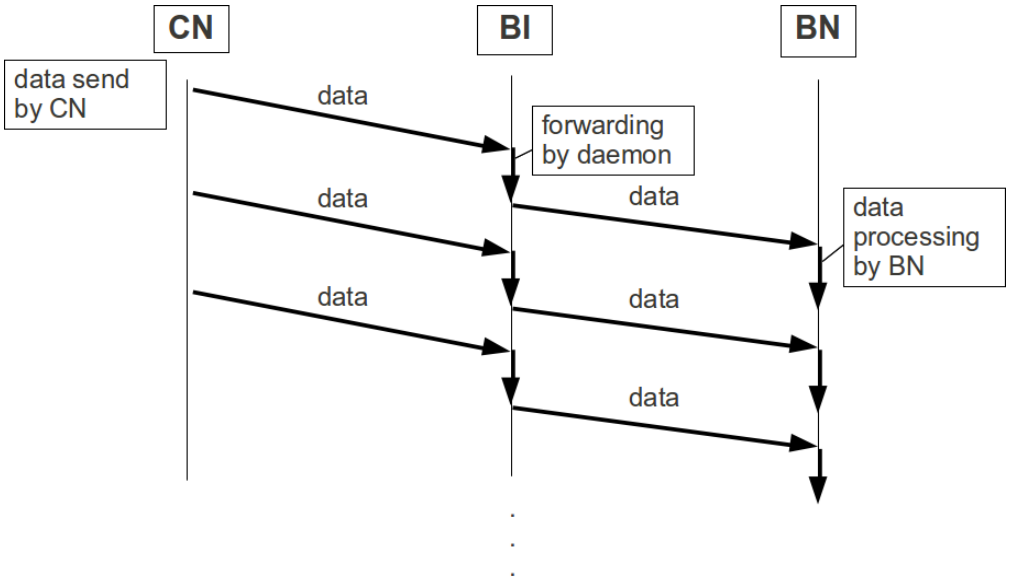

Figure 3: SW bridged write operation

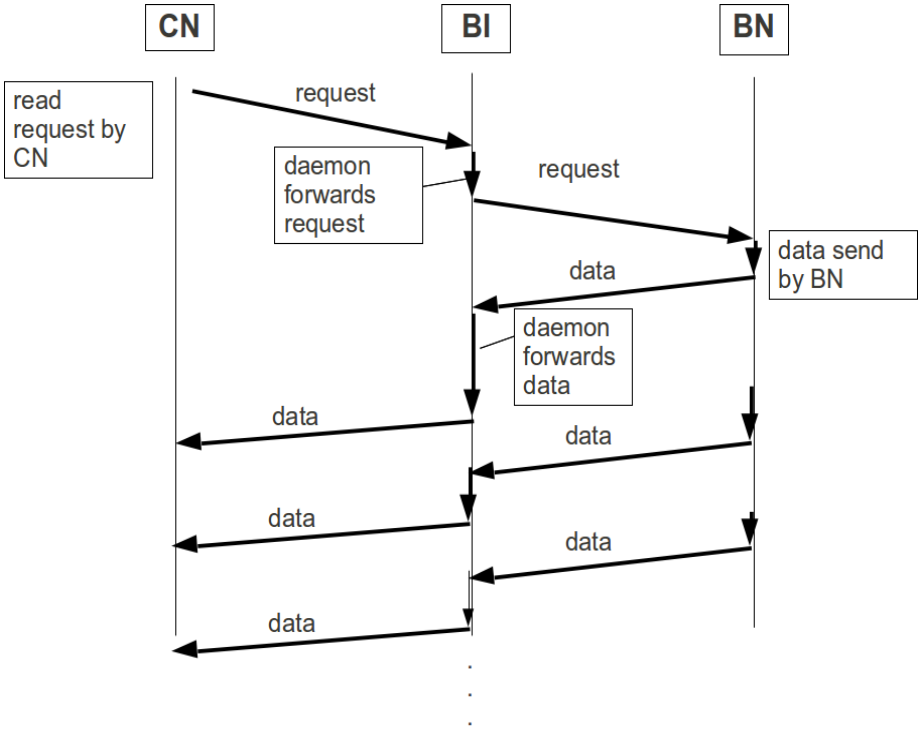

Figure 4: SW bridged read operation

3.2 IB RDMA using SMFU

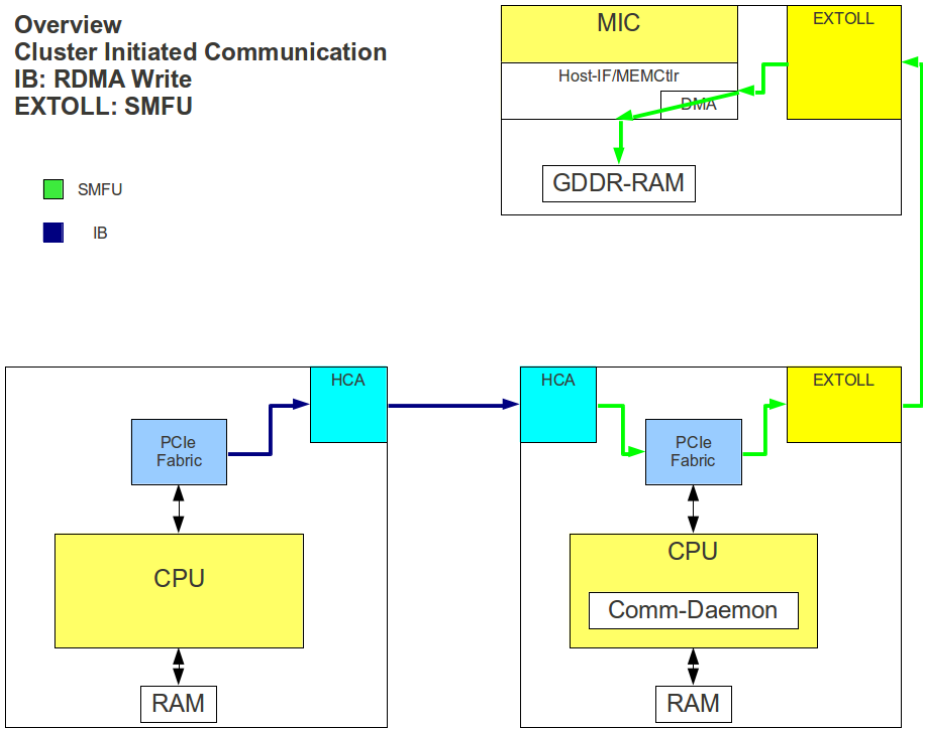

Figure 5: Write operation from CN to BN using IB-RDMA and SMFU

In order to circumvent the active involvement of the service-daemon on the BIC during the actual data-transfer, a direct communication can be obtained by leveraging EXTOLL Shared Memory Functional Unit (SMFU) functionality. The EXTOLL SMFU allows to access to a global address space spanned by that parts of the local address spaces on the BNs which are made available by this node. With that, software can remotely access the BNs memory using load/store semantics without active involvement of the BN itself. If the BIC platform is able to forward requests and responses between PCIe devices, the InfiniBand HCA can also target remote BN memory using its DMA engine. This enables for example RDMA-write and -read operations from anywhere in the InfiniBand cluster to read or write to arbitrary BN memory without active involvement of the BI host CPU.

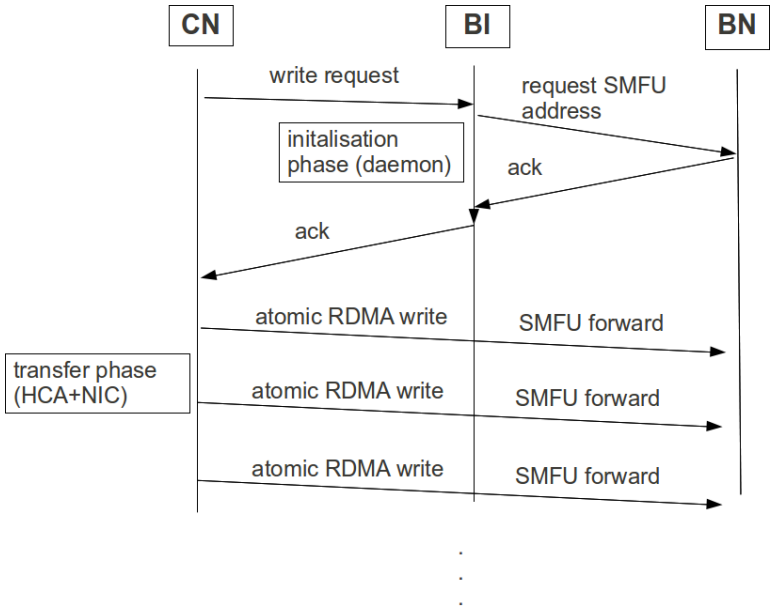

Figure 6: RDMA write operation

Figure 6 shows a sequence diagram for a RDMA-write transaction from a CN to a BN. Before the transaction can start the memory mappings on the BI must be correctly set up. In essence this means that the physical address space associated with the SMFU on the BI must be registered to the HCA driver and, thus, be available as target address for normal InfiniBand operations. A process on the CN therefore can initiate a normal RDMA-write operation to a target-address located on the BI. The write operations within the BI are transparently re-routed within the SMFU address space. On the BI all corresponding PCIe write operations are encapsulated and transported to the target BN. Here, the operation terminates and the re-mapped DMA operation is executed to the MIC's memory within the BN (Figure 7).

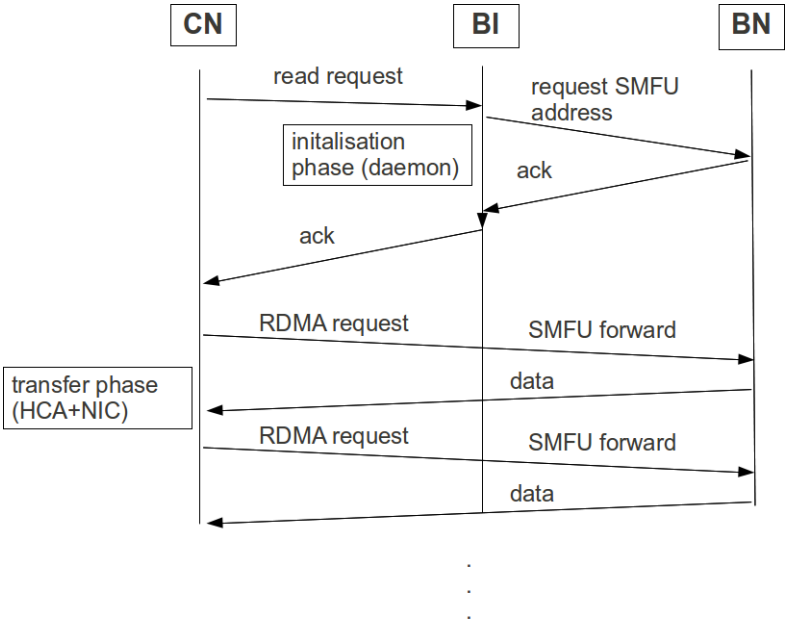

Figure 7: RDMA read operation.

The RDMA-read operation (Figure 8) works accordingly. The process on the CN posts a RDMA-read operation to the HCA in the BI. The actual PCIe reads targeted to the SMFU address space are encapsulated and forwarded to the remote BN. Here the actual PCIe memory read operation is executed and PCIe completion packets (depicted as data in Figure 8) are generated. Again, they are encapsulated and returned to the BI. The PCIe completions are forwarded to the requesting IB HCA in order to generate the corresponding IB response packets. They enable to finally complete the RDMA-read operation at the originally requesting CN.

In this scenario the duty of the service daemon on the BIC is reduced to setup the SMFU address-space and its registration to the InfiniBand HCA. During the actual data transfer no active involvement is necessary since PCIe operation targeting into the global SMFU address-space are used to translate IB-operations into encapsulated PCIe operations to be forwarded via the EXTOLL fabric.

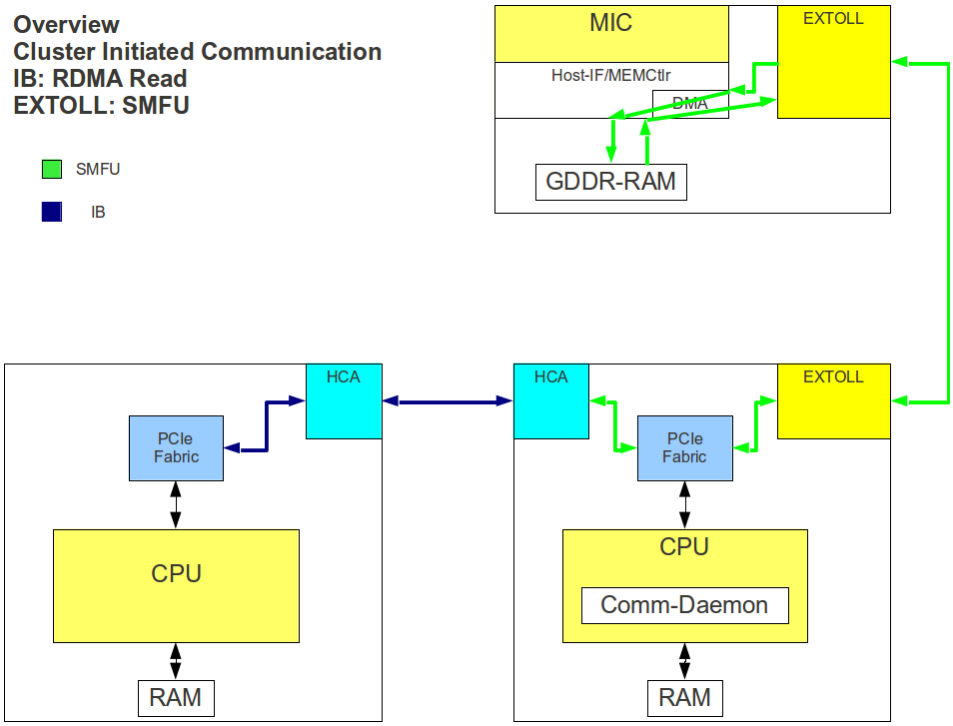

Figure 8: Read operation from CN to BN using IB-RDMA and SMFU

3.2.1 *Prerequisites for IB-RDMA over SMFU*

A first prerequisite for the active use of RDMA over SMFU is the setup of the physical memory map within the BI. This includes ensuring that a sufficiently large address-space is allocated to the SMFU. The actual realisation might depend on the capabilities of the hardware employed and will be addressed once the BIC evaluator is fully installed. In principle, there are at least two ways to do IB-RDMA over SMFU:

- Use large PCI BARs (Base Address Regions) for SMFU: If the BIC platform firmware supports multi-gigabyte bars, this is the preferred method.
- Setting up a Memory Mapped I/O (MMIO) space in the “high” address space. This is done after the system’s boot-time while the SMFU Linux kernel driver is loaded.

A second prerequisite is to give the HCA access to the SMFU address-space. Within the physical address space of the BI, this is MMIO space. Therefore the usual method used in InfiniBand drivers for “registering” memory, i.e. pinning memory and mapping into the virtual address space of a user-level process, will not work. Thus, the HCA kernel driver has to be patched in order to support the RDMA over SMFU functionality. A most compatible

and non-intrusive patch is being under investigation. One candidate is to patch that function of the driver stack which is responsible for calling the Linux API function *get_user_pages ().* Inserting a check, whether the memory to be registered is DRAM or MMIO and then to act accordingly seems to be most promising at the moment. The basic functionality has already been tested successfully with InfiniBand HCAs and an FPGA-based implementation of EXTOLL at UniHD.

Once both prerequisites are achieved, a user-space process like the proposed service-daemon running on the BI can register actual memory regions of the BN's physical memory to the SMFU address-space. It is only a matter of connection management and software to map the right regions and make the resulting keys for usage in InfiniBand available to the right processes on CNs.

3.2.2 *Direct IB-EXTOLL communication: potential problems*

Remote reads may be limited by number and size of outstanding read operations on the PCIe links between HCA and EXTOLL. The hardware currently used supports up to 256 outstanding read operations with a payload of 256 bytes each. Assuming a peak bandwidth of 8 GByte/s and dividing it through 256 outstanding operations times 256 byte payload/operation we get an estimated worst-case latency of 8 µs, which is suitable for our purposes. This issue only comes into play executing read operations since these are non-posted transactions requiring PCIe completion responses from the target. No such problem exists on the write path. Here posted transaction are used which requires no completion packets and, therefore, do not create problems due to outstanding operations.

Furthermore, the chipset of the BI must support forwarding of read and write requests. It is known that at least some chipsets only support write requests between different devices. In this case only RDMA-write operations could be implemented directly. PLX PCIe Switches support full peer to peer traffic on the PCIe fabric and are part of the BI representing server. From this a corresponding requirement on the final BIC hardware can be deduced.

The final BIC platform must also support very large PCIe Base Address Regions (BAR), in the order of multiple GB, to enable direct, flat access to the memory of a number of BNs large enough for the DEEP Booster. If the maximum BAR is limited by the firmware on the BIC, a kernel driver approach may still be possible. If the limitation results from hardware immediately (i.e. due to width of address-bus of PCIe chips), only a subset of BN memory can be directly exposed to the IB fabric. Software reconfiguration of the "aperture" can be used, but would introduce additional complexity and overhead. The current BIC evaluator supports large PCIe BARs.

3.3 Remote IB RDMA using SMFU

It is also possible to trigger RDMA-read or -write operations from BNs by accessing HCA registers directly using the SMFU. Basically the same mechanisms for memory mapping as discussed above is employed. Here the access to the HCA to start an IB operation is tunneled over SMFU/EXTOLL. While this functionality is taken into account for the software-design within this task, this solution will not be covered by this document in detail because it is part of D4.2.

3.4 Hybrid Approaches

A hybrid approach with software based translation for short packets and transparent hardware based RDMA-read and -write operations may prove best in practice. To ultimately find the right protocols, they will be evaluated by measuring the timings of read and write operations with different sizes of data on both approaches. A threshold for switching from one method to the other can then be defined in the software stack.

4 Service Daemon

4.1 Introduction

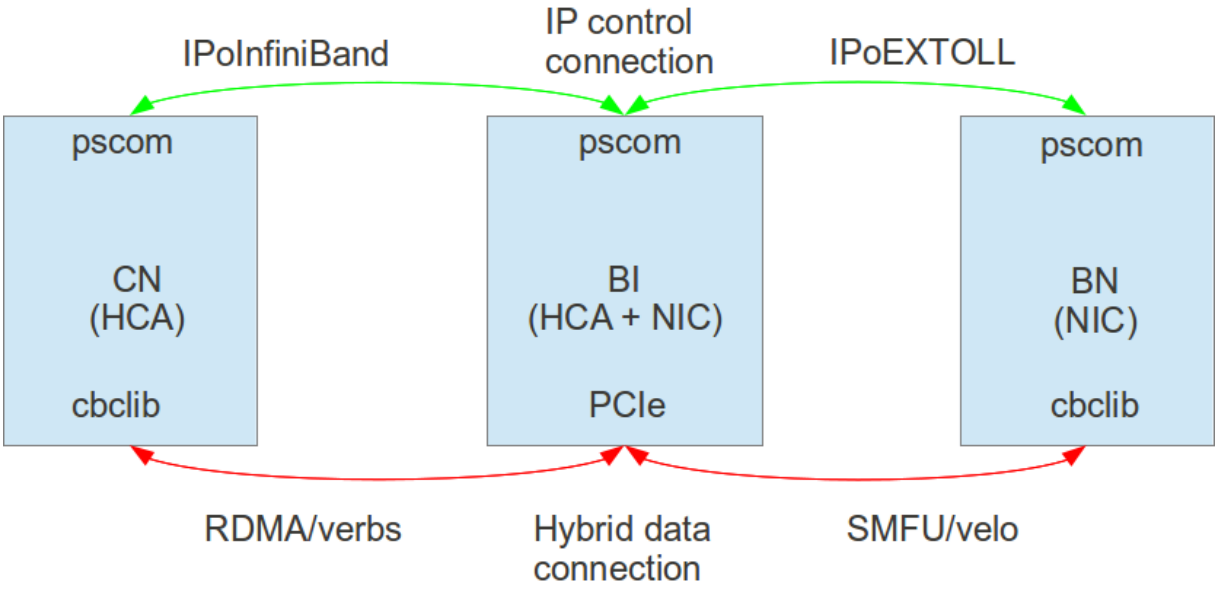

Figure 9: Sketch of control and data channel.

An actual realization of the Cluster Booster protocol will be described in this section. The original plan of the DoW intended an implementation of this protocol at the time when this deliverable is under preparation. Nevertheless, due to a delay in the procurement and bringup of the BIC evaluator, the actual implementation-work has only started recently. Therefore, only a sketch of the intended design can be presented. Instead of working on the implementation, the design considerations on the full Cluster-Booster protocol could be advances beyond the planed status of the DoW. Therefore, we are confident to be able to deliver the full Cluster-Booster protocol (D4.2) on time.

Our approach is based on two channels (Figure 9): a control channel depicted above and a data channel sketched below the three node-types. The control channel is used for protocol communication between all partners and based on a service daemon – the BI daemon – which uses ParTec's pscom library and its underlying IP stack to establish message based communication. The data channel is used for large size data transfers between CN and BN only. It is a hybrid connection established using RDMA on the InfiniBand and SMFU on the EXTOLL side as discussed in section 3.2.

4.2 Control Channel

In a first step of our approach all communication partners (CN, BN) connect to the BI daemon residing on the Booster Interface Node (Figure 10). This establishes the control channels between all involved pairs of end-points. By using the daemon as a forwarder, all participants are able to communicate with each other and the BI using the control channel, i.e. share information on how to establish a remote memory access connection. Furthermore, it can be used to exchange messages containing small amounts of data, in case the hybrid RDMA/SMFU data connection is not sufficiently performing for small messages due to large latencies originating from the complex operations required in order to setup the data-channel.

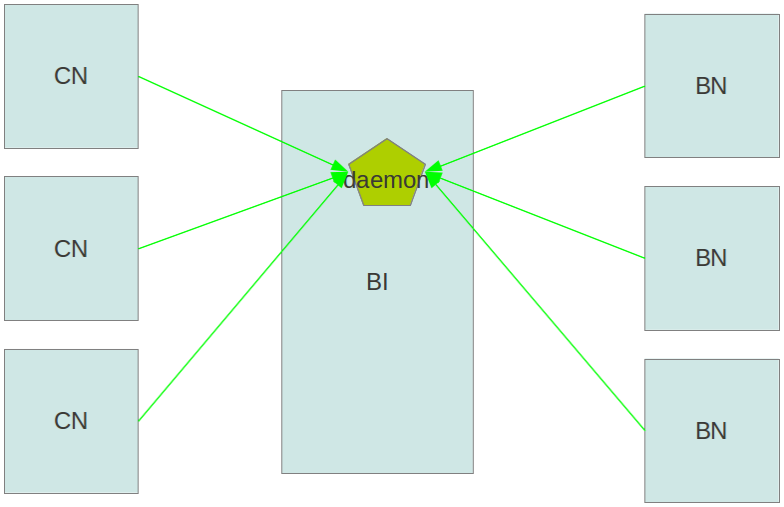

Figure 10: Step 1 – CNs and BNs are connecting to the BI using the control channel.

4.3 RDMA Data Channel

To create a RDMA data connection directly between BNs and CNs (Figure 11), all participants open an additional remote memory connection to the BI. The CNs create InfiniBand queue pairs with the BI by exchanging necessary information over the control channel. The BNs are exporting their memory regions using their SMFU units residing on the NIC to the BI, again by indicating this via the control channel.

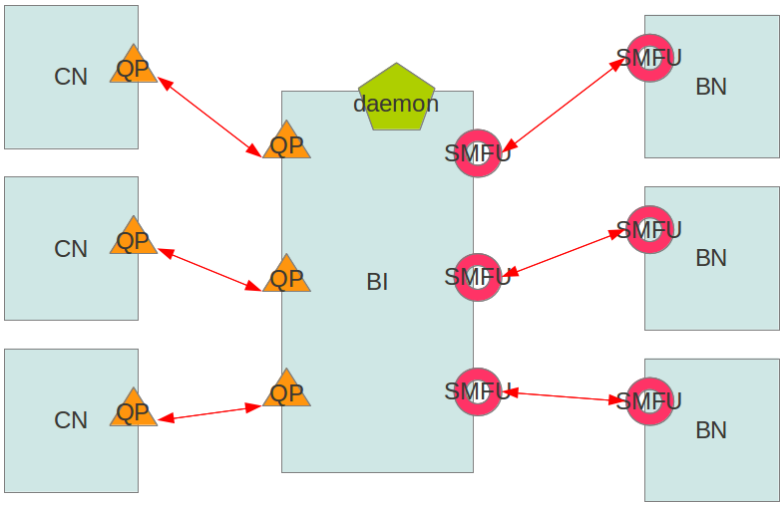

Figure 11: Step 2 – CNs establish QPairs with BI; BNs share SMFU address space with BI.

In the final step (Figure 12) a CN requests read/write access to the BNs memory by propagating this to the BI. The BI now has to register the SMFU memory region of the desired BN into the queue pair of the requesting CN. The BI then triggers the desired operation on the HCA, which transfers the data from or to the BN by reading/writing to the registered SMFU memory address.

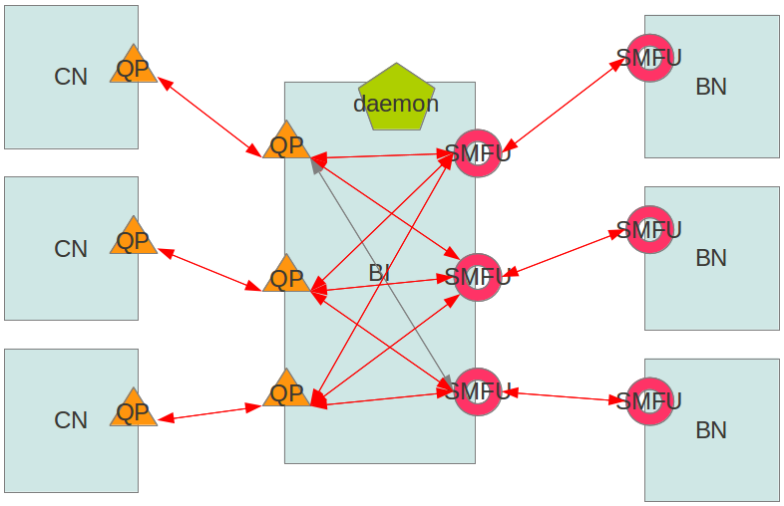

Figure 12: Step 3 – Creation of remote memory connections from CN to BN by registering SMFU address space into RDMA QPairs.

5 Expected results

The goal of this task is a high bandwidth, low overhead interconnect between an InfiniBand switched fabric and a EXTOLL 3D-torus network. From the hardware point of view both networks are connected via a PCIe link inside a special purpose platform. This enables our software stack for direct memory access from the CN side when dealing with huge amounts of data. For small data transfers a message based protocol will be implemented. The remote direct memory access can be carried out without utilizing the BIs host CPU just by triggering actions from the CN side. This ensures a high bandwidth, which is only restricted by the PCIe link inside the BI. It has to be evaluated how this strategy impacts on the latency between CN and BN. In addition, investigations will be made to find out the data size on which to switch from message based to RDMA data transfers.

References and Applicable Documents

[1] O. Pentakalos, An Introduction to the InfiniBand Architecture , 2002. [Online]. Available: http://www.oreillynet.com/pub/a/network/2002/02/04/windows.html.

[2] N. Nüssle, B. Geib, H. Fröning und U. Brüning, An FPGA-based custom high performance interconnection network., In Proceedings of the 2009 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, 2009.

[3] http://www.par-tec.com/products/parastationv5.html, ParTec GmbH, 2012. [Online].

[4] H. Fröning und H. Litz, Efficient Hardware Support for the Partitioned Global Address Space, 10th Workshop on Communication Architecture for Clusters (CAC2010), co-located with 24th International Parallel and Distributed Processing Symposium (IPDPS 2010), Atlanta, Georgia, 2012.

List of Acronyms and Abbreviations

A

API: Application Programming Interface

B

BAR: Base Address Register, registers used to announce address-mappings to PCIe devices
BI: Booster Interface (functional entity)
BIC: Booster Interface Card, interface card to connect the Booster to the Cluster InfiniBand network
BIC evaluator: Prototype system consisting of three servers representing a CN, the BI and a BN
BN: Booster Node (functional entity)
BNC: Booster Node Card is a physical instantiation of the BN
BoP: Board of Partners for the DEEP project
Booster System:Hardware subsystem of DEEP comprising of BNC, BIC and intra-Booster network

C

CN: Cluster Node (functional entity)
CPU: Central Processing Unit
CRC: Cyclic Redundancy Check

D

DDG: Design and Developer Group of the DEEP project
DEEP: Dynamical Exascale Entry Platform
DEEP Architecture: Functional architecture of DEEP (e.g. concept of an integrated Cluster Booster Architecture)
DEEP Booster: Booster part of the DEEP System
DEEP Supercomputer: A future Exascale supercomputer based on the DEEP Architecture
DEEP System: The production machine based on the DEEP architecture developed and installed by the DEEP project
DMA: Direct Memory Access

E

EC: European Commission
EU: European Union
EFlop/s: Exaflop, 10^{18} Floating point operations per second
Exascale: Computer systems or Applications, which are able to run with a performance between 10^{15} and 10^{18} floating point operations per second
EXTOLL: High speed interconnect technology for cluster computers developed by University of Heidelberg

F

FLOP: Floating point Operation

G

GPU: Graphics Processing Unit
GRS: German Research School for Simulation Sciences GmbH, Aachen and Juelich, Germany

H

HCA: Host Channel Adapter
HPC: High Performance Computing

I

IB: InfiniBand
IEEE: Institute of Electrical and Electronics Engineers
Intel: Intel GmbH Braunschweig, Germany
Interconnect evaluator: Hardware for interconnect studies on physical and mechanical layer, Developed and used in the DEEP project
IP: Internet Protocol

J

JUELICH: Forschungszentrum Jülich GmbH, Jülich, Germany

K

KNC: Knights Corner, Code name of a processor based on the MIC architecture
KNF: Knights Ferry, Intel first available processor based on the MIC

L

M

MLNX: Mellanox Technologies, Ltd., Sunnyvale, California and Yokneam, Israel
MIC: Intel Many Integrated Core architecture
MIC evaluator: Platform for evaluation of the MIC architectural concept, Used only in the DEEP project
MIC-OS: Operating System of the MIC architecture
Mini Booster prototype: Minimal instantiation of a DEEP Booster used for analysis of the energy-aware functionality, Developed and used in the DEEP project
Mini DEEP System: A fully featured DEEP System of minimal size comprising the Mini Booster
MMIO: Memory mapped input/output
MPI: Message Passing Interface, API specification typically used in parallel programs that allows processes to communicate with one another by sending and receiving messages

N

NIC: Network Interface Card, Hardware component that connects a computer to a computer network

O

OmpSs: BSC‘s Superscalar (Ss) for OpenMP
OpenMP: Open Multi-Processing, application programming interface that supports multiplatform shared memory multiprocessing
OS: Operating System

P

ParaStationMPI: Software for cluster management and control developed by ParTec
ParTec: ParTec Cluster Competence Center GmbH, Munich, Germany
PC: normally Personal Computer, but in the context of the proposal also Project Coordinator
PCI: Peripheral Component Interconnect, Computer bus for attaching hardware devices in a computer
PCIe: PCI Express, Standard for peripheral interconnect developed to replace the old standards PCI, improving their performance
PCIe root complex: Connects CPU and memory to the PCIe switch fabric. It includes functionality for scanning the PCIe topology and initialisation of PCIe devices.

PFlop/s: Petaflop, 10^{15} Floating point operations per second
PLX: Company building PCIe switches
PM: Project Manager of the DEEP project
PMT: Project Management Team of the DEEP project
PR: Public Relations
Project Coordinator: Leading scientist coordinating and representing the DEEP project

Q

QDR: Quad Data Rate, communication signalling technique of InfiniBand
QP: Queue Pair, communication channel defined within the InfiniBand standard

R

RDMA: Remote Direct Memory Access
R&D: Research and Development

S

SC: International Conference for High Performance Computing, Networking, Storage, and Analysis, organized in the USA by the Association for Computing Machinery (ACM) and the IEEE Computer Society
SMFU: Shared Memory Functional Unit
SW: Software

T

TFlop/s: Teraflop, 10^{12} Floating point operations per second
TK: Task, Followed by a number, term to designate a task inside a work package of the DEEP project
ToW: Team of Work Package leaders within the DEEP project

U

UniHD: University of Heidelberg, Germany

V

W

WP: Work Package

X

x86: Family of instruction set architectures based on the Intel 8086 CPU

Y

Z

ZITI Heidelberg: Institut für Technische Informatik Uni Heidelberg, Germany