Integration of Intel Xeon Phi Servers into the HLRN-III Complex: Experiences, Performance and Lessons Learned

Florian Wende, Guido Laubender, and Thomas Steinke
Zuse Institute Berlin (ZIB)
Berlin, Germany
Email: {wende, laubender, steinke}@zib.de

***Abstract*—The third generation of the North German Supercomputing Alliance (HLRN) compute and storage facilities comprises a Cray XC30 architecture with exclusively Intel Ivy Bridge compute nodes. In the second phase, scheduled for November 2014, the HLRN-III configuration will undergo a substantial upgrade together with the option of integrating accelerator nodes into the system. To support the decision-making process, a four-node Intel Xeon Phi cluster is integrated into the present HLRN-III infrastructure at ZIB. This integration includes user/project management, file system access and job management via the HLRN-III batch system. For selected workloads, in-depth analysis, migration and optimization work on Xeon Phi is in progress. We will report our experiences and lessons learned within the Xeon Phi installation and integration process. For selected examples, initial results of the application evaluation on the Xeon Phi cluster platform will be discussed.**

***Keywords*-K.6.2.d Performance and usage measurement, K.6.4 System Management, Xeon Phi cluster, system integration**

I. Introduction

The Cray XC30 system installed at ZIB represents together with its sister installation in Hannover the third generation of the North German Supercomputing Alliance (HLRN) compute and storage facilities. The current Cray XC30 configuration of phase I is homogeneous with exclusively Intel Ivy Bridge compute nodes.

Within the next years, advances in manufacturing processes continue and enable processor architectures with increasing counts of powerful compute cores (many-core processors) and integrated infrastructure for attaching memory and networks devices at moderate power envelopes and pricing. The XC30 architecture has at its heart an interconnect infrastructure which enables scalability at today's application requirements level. Furthermore, many-core devices (Nvidia GPU and Intel MIC) can be integrated to build a heterogeneous system which is attractive for certain compute-bound applications with sufficient parallelism. We believe that within the next years, heterogeneous compute and storage architectures will become more and more mainstream in HPC to leverage compute and storage resources without sacrificing the power envelope as visible in the TOP500 [4].

Research in Many-Core High-Performance Computing

At ZIB, recently the Research Center "Many-Core High-Performance Computing" being one of Intel Parallel Computing Centers (IPCC) world-wide was founded. This research center aims at porting and improving HPC workloads on many-core technologies. Additionally, we provide best-practice experiences for many-core code developers and we develop low-level system software for, e.g., faster upload of HPC code on many-core devices.

With respect to the envisaged growing importance of many-core computing in HPC, we evaluated in the past disruptive processing technologies, including FPGAs, ClearSpeed, and Cell BE processors [17], [21], [22], [24], as well as general-purpose GPUs [7], [23], [27], and recently the MIC architecture [28].

To foster the evaluation of many-core processors for workloads on the HLRN system, a four-node Intel Xeon Phi cluster is integrated into the present HLRN-III infrastructure at ZIB.

The structure of this paper is as follows:
Section II describes the overall HLRN-III configuration and in particular the installation at ZIB with its services. Sections III and IV presents the configuration of the Xeon Phi test cluster and its integration into the HLRN infrastructure. In section V we discuss selected results of our migration and optimization work on Xeon Phi. We summarize our experiences and lessons learned within the Xeon Phi installation and integration process in Sect. VI.

II. HLRN-III Cray XC30 Installation

As its predecessor systems HLRN-I and HLRN-II, the HLRN-III system is distributed among two sites and is operated at Zuse Institute Berlin (ZIB) and at Leibniz University IT Services (LUIS) in Hannover.

A. Hardware Configuration

In the first installation phase (since fall 2013), a Cray XC30 system with exclusively Intel x86 nodes provides high-end compute power (1488 nodes in total with two 12-core Intel IvyBridge CPUs each). At the Hannover site, 32

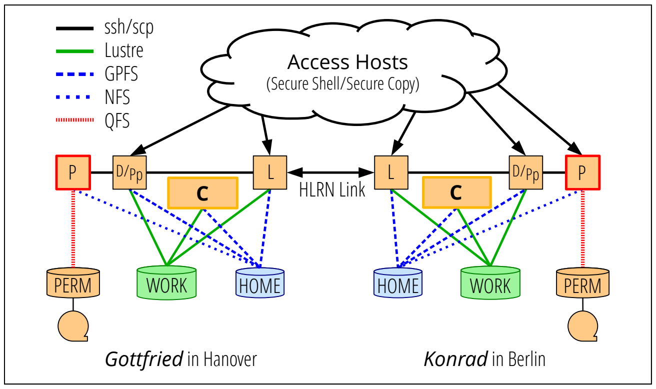

Figure 1. Schematic representation of the distributed HLRN-III system. C: compute system consisting of Cray XC30 and MEGware SMP nodes; L: login nodes, D: data nodes, Pp: postprocessing nodes; P: archive servers.

Figure 2. The Cray XC30 system “Konrad” in Berlin (top) in its final stage will comprise 10 cabinets whereas the XC30 system ”Gottfried” in Hannover (bottom) will offer 9 cabinets and 64 additional external SMP nodes in total.

additional large memory quad-socket SMP nodes with Intel octa-core SandyBridge processors are part of the HLRN-III infrastructure.

Figure 1 shows a schematic representation of the distributed HLRN-III system with its two sites Berlin and Hannover. The two sites are connected by a fast dedicated fiber optical link (10 Gbps) which is used to provide a single-system view (single login) to the user.

Besides the usual login nodes for user access the system is also equipped with additional service nodes like data mover and pre- and postprocessing nodes with large memory on both sites, or services to access the archive systems.

For the per site globally accessible on-line storage two different technical solutions are in use. The HOME file system (total capacity of 1.4 PB) is realized as a network attached storage (NAS) from DataDirectNetworks (DDN) whereas the parallel file system for fast I/O in batch jobs is implemented with a Cray Lustre File System (CLFS) with a total capacity of 2.8 PB in the first phase. The data is stored on a DDN block storage systems.

As archival storage for permanent data sets Oracle StorageTek tape libraries with a capacity of several PB (depending on the user requirements) and nearly 100 TB on-line disk caches are additionally provided on each site.

In the second phase, scheduled for fall 2014, the HLRN-III will undergo a substantial upgrade of the XC30 system on both sites together with the option of integrating accelerators like Intel Xeon Phi or GPGPUs into the Cray XC30 system (Fig. 2).

B. *Batch Service and Programming Environment*

For the job scheduling and accounting Moab/Torque is implemented. If the Moab Grid Scheduler is fully in operation the HLRN-III can provide a single point of view to the user in terms of handling jobs. Job accounting is realized through the MAM component of Moab and is capable of providing a single accounting instance today for all jobs on the HLRN-III system. This includes jobs run on the MPP (Cray XC30) and SMP nodes, the data movers, pre-/post-processing nodes and (as described below) on the Xeon Phi nodes. Different weighting factors are applied to job charging for each of different node types.

The Programming Environment is based on Cray versions of the compiler, libraries and performance analysis tools. It is supplemented by domain specific libraries and tools installed and maintained by HLRN staff. A basic tool for version management is the `modules environment` package.

C. *HLRN-III Workloads*

HLRN has a broad user community in the universities and scientific institutions of the seven participating North-German federal states. A wide palette of topics in the specific scientific areas of the HLRN clients needs to be supported. The major scientific fields are chemistry and material science, earth sciences, engineering, and physics. Thus, the HLRN-III system has to support a broad range of HPC applications efficiently as possible across the various scientific disciplines.

The program packages used on HLRN-III are mostly academic and developed by community members. Major compute cycles are consumed by BQCD [18] and FRESCO [1] (both physics), CP2K [25] (chemistry and material sciences), OpenFOAM [3] (engineering), PALM and PALM/particle [20], NEMO [2] and FESOM [26] (all geosciences), which were part of the HLRN-III application benchmark suite in the procurement.

III. The Integrated Xeon Phi Test Cluster

The current HLRN-III complex at ZIB contains a separate but fully integrated four-node Intel Xeon Phi cluster. With the many-core upgrade of HLRN-III in mind, the Xeon Phi cluster serves for the HLRN-III users as a development and testing system for code portation to the Xeon Phi, and for the HLRN-III administrators as a training assistance to

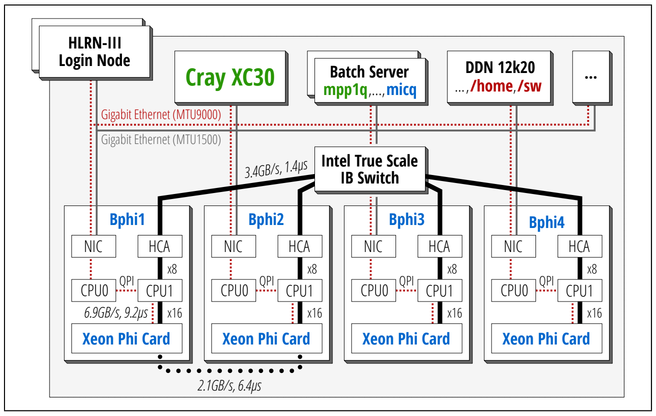

Figure 3. Four-node Xeon Phi cluster (nodes: Bphi[1-4]) as a separate but fully integrated component of the HLRN-III installation consisting of the Cray XC30 and the attached user and administrative infrastructure. Bphi[1-4] spawn an independent InfiniBand subnetwork using Intel True Scale fabric components, and are connected to HLRN-III via Gigabit Ethernet.

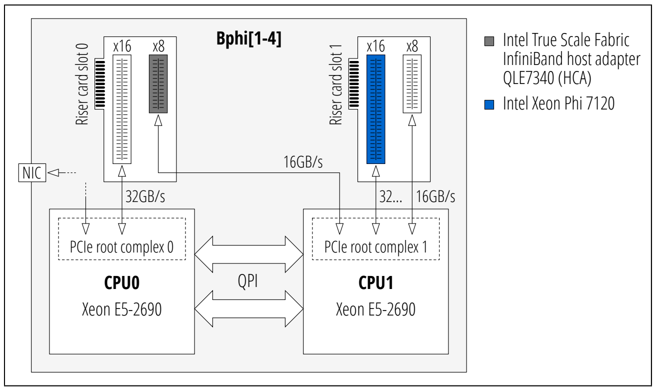

Figure 4. Assignment of PCIe card slots to PCIe root complexes of the two Intel Xeon E5 CPUs. Both the Xeon Phi card and the InfiniBand HCA are placed into PCIe slots attached to the same PCIe root complex.

become acquainted with the integration and management of the Xeon Phi into the HLRN-III infrastructure. Further, the cluster is used by ZIB for code development throughout its many-core research center activities.

Each node of the test cluster contains two Intel Xeon octa-core CPUs and hosts one Intel Xeon Phi coprocessor. The four nodes spawn an independent InfiniBand (IB) subnetwork which allows for fast communication between the nodes and the Xeon Phi coprocessors.

Figure 3 illustrates the integration of the Xeon Phi nodes into HLRN-III complex in Berlin. The network connection to the HLRN-III service nodes and storage resources is realized via Gigabit Ethernet. The nodes reside in the same subnet as the XC30 MOM nodes.

It follows detailed information on the hardware and the software configuration of the Xeon Phi nodes.

A. *Hardware Configuration of the Phi Cluster*

The compute nodes each hold two Intel Xeon E5-2690 Sandy-Bridge octa-core CPUs (clocked @2.9 GHz, Hyper-Threading enabled) that are installed into an Intel server board S2600GZ [15]. Each CPU socket has attached 12 DDR3 DIMM slots that are fully populated with 4 GB DIMMs for a total of 96 GB main memory.

The system includes two riser card slots on the server board, where riser card slot 0 is attached to CPU 0 and riser card slot 1 to CPU 1. Our installation uses riser cards with one PCIe x16 Gen3 and one PCIe x8 Gen3 slot each. Per node one Intel Xeon Phi 7120 coprocessor card [13] (61 physical cores with 4-way hardware multi-threading each, and 16 GB main memory) and one single-port Intel True Scale Fabric InfiniBand QLE7340 HCA [12] (3.4 GB/s unidirectional throughput and latency down to 1.0 μs) are installed into the PCIe slots. To ensure direct access to the HCA from the Xeon Phi, it is necessary to have both of the two attached to the same PCIe root complex. For the S2600GZ server board the Xeon Phi thus need to be placed into the x16 slot of riser card 1, and the HCA in any of the remaining PCIe x8 slots. The setup is schematically illustrated in Fig. 4.

The Xeon Phi nodes are connected with each other via an Intel True Scale Fabric 12300-BS01 switch that provides 36 4x QDR ports with 4 GB/s throughput per port [11].

Table I summarized the performance of our True Scale InfiniBand network. All values have been determined with the Intel MPI benchmark 3.2.4. The respective maximum transferrates and minimum latencies are noted in Fig. 3.

Table I
TRANSFERRATES (GB/S) AND LATENCIES (μS) FOR COMMUNICATING MPI RANKS (A) ON TWO DIFFERENT HOST NODES, (B) ON ANY OF THE HOST NODES AND THE XEON PHI INSTALLED WITHIN THAT NODE, AND (C) ON TWO XEON PHI CARDS WITHIN TWO DIFFERENT HOST NODES.

	Fabric	# Ranks	Transferrate	Latency
(A) Host-to-Host	TMI	2	1.8 GB/s	1.4 μs
	TMI	4	3.4 GB/s	1.5 μs
	TMI	8	3.0 GB/s	4.5 μs
	TMI	16	3.0 GB/s	8.0 μs
(B) Host-to-Xeon Phi	SCIF	2	5.7 GB/s	9.2 μs
	SCIF	4	6.5 GB/s	12.6 μs
	SCIF	8	6.8 GB/s	41.0 μs
	SCIF	16	6.9 GB/s	62.0 μs
(C) Xeon Phi-to-Xeon Phi	TMI	2	0.4 GB/s	6.4 μs
	TMI	4	0.8 GB/s	7.1 μs
	TMI	8	1.5 GB/s	7.5 μs
	TMI	16	2.1 GB/s	9.3 μs

B. *OS Configuration & Xeon Phi Setup*

All four Xeon Phi nodes run a locally installed CentOS 6.3 64-bit Linux operating system with kernel version 2.6.32-279, as recommended by Intel at the time of the installation [9], [14]. Our first attempts to use CentOS 6.4 instead resulted in difficulties regarding Xeon Phi specific software components with only little official support by Intel to resolve them (at this time).

For the basic OS installation we excluded all InfiniBand components provided by CentOS 6.3 in favor of the Intel proprietary Infiniband driver package for True Scale HCAs (version 7.2.0.0.42).

The integration of the Xeon Phi coprocessor into the CentOS 6.3 host requires the installation of the Intel Many-core Platform Software Stack (MPSS for short) which amongst others

- provides a Linux based μOS for the coprocessor,
- introduces the SCIF (Symmetric Communication Interface) driver layer with SCIF being the communication backbone between the host and the coprocessor.
- supports standard compliances like sockets, TCP/UDP IP (over PCIe), PSM, OFED Verbs, MPI, OpenMP, etc.

We use the MPSS 2.1 release (version 2.1.6720-19). On a separate Xeon Phi node we tested the current MPSS 3.1 release but found no siginificant performance improvements over MPSS 2.1. Hence, updating from MPSS 2.1 to MPSS 3.1 appears not meaningful to us.

Network Configuration

The initial network configuration includes the creation of network interfaces for the Phi cards, the installation of the OFED stack and loading of proper IB RDMA kernel modules on the Phi system. Details of this installation phase are described in Appendix B.

In the next step, the system components for the communication between all OS instances, i. e. host nodes and their Phi nodes inside are configured. This includes the definition of sub-networks, routing, IP forwarding and NAT. Details can be found in Appendix C.

Infiniband, Intel MPI & the TMI Fabric

For communication across the cluster hosts and the Xeon Phi cards a private Intel True Scale InfiniBand sub-network is configured. As communication protocol Intel MPI 4.1.1 and 4.1.3 is provided together with the Intel TMI fabric for tag matching interfaces such as True Scale.

Running MPI programs on the subcluster using the TMI fabric is done e. g. via

```
# mpirun -psm ... ./proc.x
```

For that, on both the host and Xeon Phi system TMI needs to be configured properly (Appendix D).

The TMI fabric works across our entire Xeon Phi cluster with the limitation that per node either the host or the Xeon Phi can be used, but not both of them simultaneosuly. For the execution of MPI programs utilizing the host and the Xeon Phi card attached to it, SCIF needs to be selected as fabric e. g. through the `-env I_MPI_FABRICS=dapl` argument when calling `mpirun`.

With the MPSS 3.3 the True Scale "native mode" will be supported which should remove this drawback.

IV. Integrated Resources of the HLRN-III Complex

To support a smooth utilization path to the Xeon Phi resources for the HLRN users, the Phi test cluster integrates

- access to HLRN user's HOME file systems and central software repository,
- the user and project management, and
- the job management and resource accounting.

A. Access to HLRN file systems

Both the host nodes and the Xeon Phi cards mount the HLRN-III user's HOME directory and the HLRN's central software repository to `/home/b` respectively `/sw`.

For NFS mount to work on Xeon Phi, for our setup the aforementioned NAT have been applied. In order to create the mount points for the above directories when (re)starting the MPSS service, the filelist configuration for the Phi card must be updated (Appendix E).

B. Software & Development Tools

As for all HLRN nodes the `module environment` package is installed on the Phi cluster nodes to take advantage of the central HLRN software repository maintained by the HLRN staff. In this way, the Intel development and analysis suite including Intel compiler, MKL libraries and performance analysis tools (VTune, ITAC) is provided to the user through their installation in `/sw`. Additionally, the GNU compiler suite (version 4.8.1) and PAPI 5.3.0 are installed, for example.

For giving Intel VTune access to the hardware performance counter the SEP kernel module are installed both on the cluster host and the Phi card.

C. User and Project Management

Access to Phi test cluster is authorized by the central HLRN LDAP service. Since the OS version level supported by the Intel MPSS is not up-to-date and to lift the security pressure from the administrators, the Phi host nodes are only accessible from the HLRN login nodes.

Resource accounting on the Phi cluster nodes is implemented through the HLRN job management and accounting system (see the following subsection). The account of a user's project is charged with a weighted factor of the utilized wall-clock time.

D. Batch System Configuration for Xeon Phi

On the HLRN-III, Torque and Moab from Adaptive Computing are used as resource manager and scheduler in the batch system (the currently stable running versions are Torque 4.2.6 and Moab 7.2.6). To enable communication with the Moab/Torque batch system, the Phi nodes reside in the same subnet as the XC30 MOM nodes and can be allocated from the HLRN-III user login nodes via the batch system.

Torque's pbs_mom daemon running on the Phi nodes are compiled with MIC support, so that the Phi devices are automatically detected by Torque and known to the scheduler. Additionally, a special MIC class called "micq" was configured in Moab for the Phi cluster nodes, and a node feature called "mic" is configured in Moab.

As the currently used MPSS version 2.1 is lacking of direct LDAP support, the LDAP userid of the batch job owner is dynamically added to the μOS of the Xeon Phi in a job prologue script and removed in a corresponding epilogue script.

For MPI jobs running on the Xeon Phi the PBS_NODEFILE needs to be manually modified inside the job script by adding the hostname(s) of the Phi card(s).

V. Case Studies using the Xeon Phi

In the context of the activities of the IPCC at ZIB and in collaboration with HLRN program developers, selected HLRN and ZIB workloads are evaluated and optimized on the Xeon Phi platform. The selected workloads cover various scientific areas from quantum chromodynamics (BQCD), material science (VASP, 2D/3D Ising models), thermodynamical sampling of molecules (GLAT), turbulent flow simulation (PALM) or the simulation of photoactive biomolecules (GPU-HEOM).

In the following, we present results of our evaluation and performance optimization on Xeon Phi for the two workloads BQCD and Ising 2D/3D.

A. *BQCD – Berlin Quantum ChromoDynamics*

BQCD [18] is a Hybrid Monte-Carlo program for the simulation of lattice QCD with dynamical Wilson fermions. The program can simulate 2 and 2+1 fermion flavours with pure, clover improved, and stout smeared fat link Wilson fermions as well as standard plaquette, and an improved (rectangle) gauge action. The single flavour is simulated with the Rational Hybrid Monte-Carlo algorithm.

The development of BQCD by H. Stüben (and later also by Y. Nakamura) started in 1998 at ZIB. Since 2010 BQCD is under the GNU General Public License. At ZIB, the BQCD program has been used as benchmarking code for the Cray XC30 supercomputer due to its well known strong scaling behavior to thousands of compute cores.

BQCD is currently extended through the libqcd library developed by Th. Schütt at ZIB to make use of the Intel Xeon Phi coprocessor for the execution of the conjugate gradient (CG) method—CG is a core element of BQCD. Executing CG on Xeon Phi from within the original Fortran 77 program goes through libqcd's Fortran interface which is written in C++ and uses Intel's Language Extension for Offload (LEO).

The implementation of the CG method on the Xeon Phi differs from its original Fortran77 counterpart in that the memory layout has changes from Array-of-Structure (AoS) style to an Array-of-Structures-of-Arrays (AoSoA) style to better support SIMD operations. The Xeon Phi CG-kernel has been implemented using SIMD intrinsic operations.

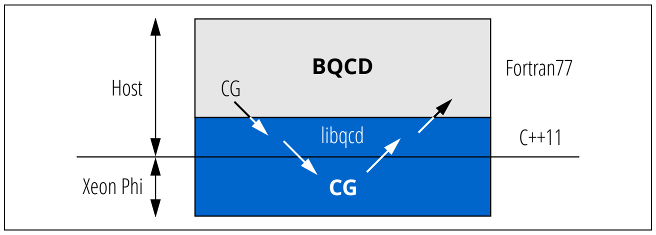

Figure 5. Schematic illustration of the libqcd architecture.

Fig. 6 illustrates the performance of the CG kernel on both CPU host and Xeon Phi for a $32\times32\times32\times64$ lattice. On Xeon Phi 240 OpenMP threads are used for the execution, whereas on CPU 16 (one CPU socket) respectively 32 (two CPU sockets) OpenMP threads are used. Compared against the two-socket CPU execution, a speedup of about 1.75 can be achieved by offloading the computation to the Xeon Phi.

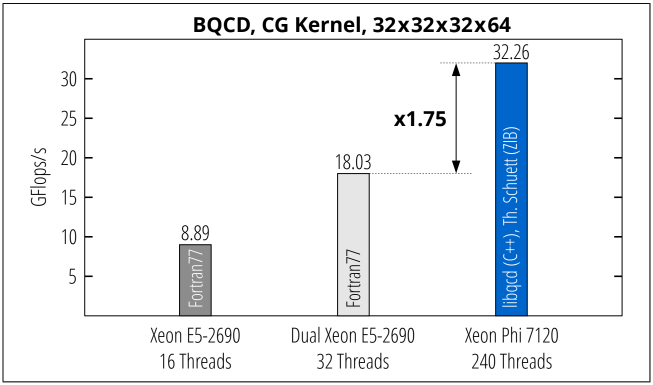

Figure 6. Performance of the BQCD CG kernel in GFlops/s obtained on a single node of the Xeon Phi test cluster.

B. *Swendsen-Wang Cluster Algorithm for the Ising Model*

The Ising model is a simple spin model describing ferromagnetism in Statistical Physics. The Ising model postulates a d-dimensional regular lattice with spins s_i placed on the lattice sites and pointing either in the one or in the opposite direction in space (represented e.g. by spins take on values $s_i = \pm1$). The spins interact locally with their nearest neighbors according to the Hamiltonian $H = -\sum_{\langle ij\rangle} J_{ij}s_is_j + b\sum_i s_i$, where $\langle ij\rangle$ denotes spin s_i and s_j are nearest neighbors, J_{ij} is a coupling constant, and b is an external magnetic field.

Due to the enormous number of spin configurations s^μ on "large" lattices (2^N for N lattice sites), simulations of the Ising model on the computer are usually done by means of Monte-Carlo methods, where each configuration is generated with the probability $p_\mu = \mathrm{e}^{-E_\mu/kT}\left(\sum_\mu \mathrm{e}^{-E_\mu/kT}\right)^{-1}$ of its

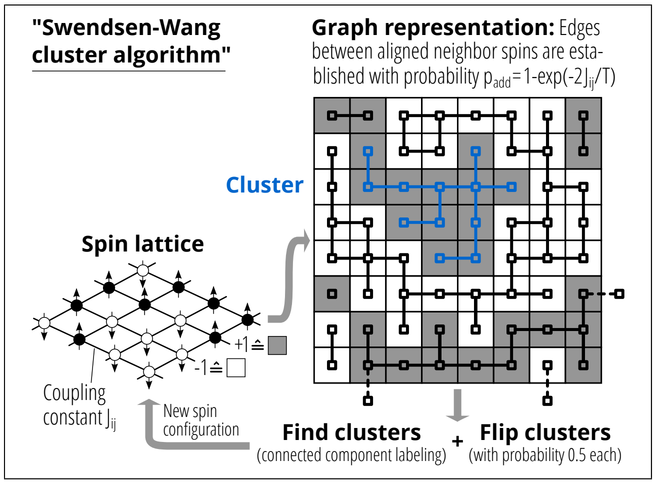
"Swendsen-Wang cluster algorithm"
Graph representation: Edges between aligned neighbor spins are established with probability p_{add}=1-exp(-2J_{ij}/T)
Cluster
Spin lattice
+1≙
-1≙
Coupling constant J_{ij}
New spin configuration
Find clusters
(connected component labeling)
+
Flip clusters
(with probability 0.5 each)

Figure 7. Schematic illustration of the Swendsen-Wang cluster algorithm. Clusters correspond to regions of aligned, edge-connected neighboring spins. Finding the clusters maps onto the connected component labeling problem. After having found all clusters, each of them is flipped over with probability 0.5 in order to create a new spin configuration.

occurence—T is the temperature at which the simulation takes place.

For $d \geq 2$, and with $b = 0$, the Ising model exhibits a phase transition; there is a high-temperature phase with spins oriented almost at random, and a low-temperature phase with almost almost all spins pointing. The two phases are separated by the "critical temperature" T_c. Close to criticality spins organize into clusters, which are regions of neighboring spins pointing in the same direction. Monte-Carlo simulations of the Ising model at criticality are most efficient by means of cluster algorithms which abstract the spin system as being made up of clusters and apply changes to the clusters as a whole to move the spin system from one configuration to another.

We implemented the Swendsen-Wang cluster algorithm (Fig. 7 for illustration) on both x86 CPU and Intel Xeon Phi using MPI for inter-node communication and OpenMP to achieve parallelism on the nodes—we use a two-layer domain decomposition to define independent problems that can be solved in parallel. At the core of the Swendsen-Wang algorithm is "connected component labeling" which on the Xeon Phi has been implemented using SIMD instrinsic operations—we create native executables on Xeon Phi. On the CPU vectorization is left to the compiler. A detailed description of the algorithm and its implementation can be found in [28].

Performance values measured on the Xeon Phi test cluster at HLRN-III are given in Fig. 8 for selected two- and three-dimensional regular lattices. We use one MPI rank per Xeon Phi device respectively CPU host. We use 240 OpenMP threads on Xeon Phi and 16 OpenMP threads on the host (pinned to one CPU socket, with Hyper-threading enabled). MPI communication is directly between MPI ranks

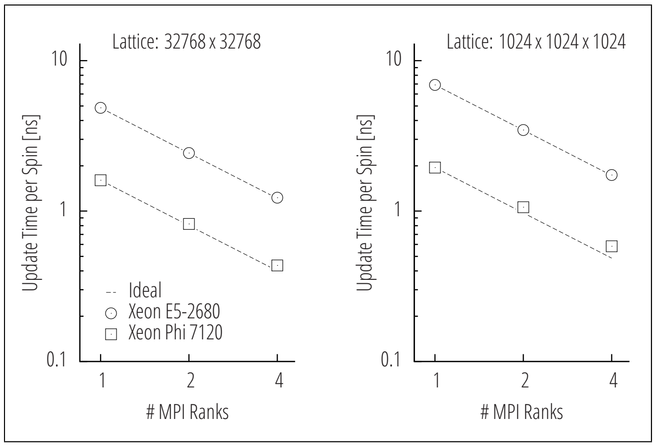

Figure 8. Performance of the Swendsen-Wang cluster algorithm on the Xeon Phi test cluster at HLRN-III. The MPI and the threading setup is described in the text.

on Xeon Phi (native executable) and CPU host, respectively.

According to Fig. 8, in both two and three dimensions speedups of about a factor 3 can be achieved over the CPU when using the Xeon Phi. The strong scaling of the implementation on CPU is almost ideal, whereas on Xeon Phi a little performance decrease can be observed—we found our implementation using MPI be bound by the latency of the network used for inter-node communication.

VI. Xeon Phi Cluster Integration: Lessons Learned

We will summarize our lessons learned during the installation and integration of Intel Xeon Phi cluster into the HLRN-III complex.

Timeline: Prior the installation of the Xeon Phi cluster, the HLRN-III complex with the Cray XC30 system and the various service nodes was installed and configured by Cray personnel. During the network configuration of the Phi cluster the internal HLRN networks underwent optimizations, e. g. changing MTU sizes which impacted the usability of the Phi nodes (i. e. on *all* network devices along the data path from the storage server to the Phi nodes the same MTU size must be configured). Due to time and personal restrictions it was decided not to give access to the HLRN Lustre file system from the Phi nodes.

Batch system: The integration of the Phi nodes into the Moab/Torque batch and scheduling system is subject to the same restrictions as for any node, in particular the ability of reverse name resolution for the communication between batch server and client process is required. To enable the visibility of the MIC resources automatically, the Torque client daemon must be re-compiled with the proper configuration option `--enable-mics` [5]. Phi nodes and the "mic" class and feature in Moab can be integrated without side-effects into the site-wide batch configuration during production.

Since our current MPSS version does not support LDAP on the OS of the Phi card itself we development a workaround for enabling temporary users on the Phi μOS. By means of the Torque prologue script the LDAP user ID is validated and added to the user accounts on the Phi μOS, whereas the epilogue script removes the user ID from the Phi μOS.

LDAP: The LDAP service is managed by the local system administrator group, and their procedure for configuring the LDAP client on the CentOS based Phi nodes went smoothly.

Security aspects: Due to the MPSS support for old (CentOS) kernels the access to the Phi cluster is not open from the public external network. Only authorized HLRN users can login into the Phi nodes from the HLRN login service nodes or can submit batch jobs on the Phi cluster.

Phi node hardware and network configuration: Because of the small number of Phi nodes we decided to implement a flat network for the (MPI) communication between the Phi host nodes and the Phi cards. This brings in a two-fold complexity of the network configuration coming with the Xeon Phi card being a self-hosting OS instance. Although the Intel documentation provides a valuable source of information for setting up a Phi cluster we were in particular surprised by the initial bad communication performance. In discussions with Intel experts the importance that the Xeon Phi card and IB NIC needs to be in the same PCIe root complex was outlined.

APPENDIX A. PROLOGUE AND EPILOGUE SCRIPTS FOR THE USER ID TRANSFER

Torque's ability to run scripts before and after a batch job execution [6] is used to temporarily add a LDAP user ID from the host environment to the Phi μOS in a job prologue script and to remove it after job termination in a job epilogue script.

As first step in our prologue script the current `passwd` and `shadow` files of the Phi μOS are copied to the hosts filesystem as a backup to be able to restore them later on.

In a second step the output of the command on the host for the specific user ID of the batch job owner is added to the `passwd` file of the Phi μOS (by LDAP **`getent passwd $userID`**).

Also the user entry (`$userID:⋆:::::::`) is added to the `shadow` file of the Phi μOS so that the LDAP user ID is finally known on the Phi.

In the epilogue script the beforehand saved `passwd` and `shadow` files of the Phi μOS are transfered back on to the Phi so that the user account which was added in the prologue script is removed from the Phi μOS. Finally, the Phi μOS is rebooted to provide a well defined state of the Phi card for new jobs.

APPENDIX B. INITIAL NETWORK AND DRIVER SETUP

After the installation of MPSS, the execution of

```
# micctrl --initdefaults
```

for each Xeon Phi card creates a network interface `micX` on the host system for communication with card `X`. Further, for each card a subdirectory `INTEL_INSTALL_PATH/mic/filesystem/micX` is created, containing configuration files and directories `/etc`, `/home`, etc. In our case the interface `mic0` is created, and the card is added to `/etc/hosts` with alias `mic0`.

Along with the installation of MPSS we installed OFED (OpenFabrics Enterprise Distribution) for Xeon Phi. Both the MPSS and the OFED stack are started via init-scripts,

```
# service mpss start
Starting MPSS Stack:                  [ OK ]
mic0: online (mode: linux image: ...)
# service ofed-mic start
Starting OFED Stack:
host                                  [ OK ]
mic0                                  [ OK ]
```

After starting the OFED layer on the Xeon Phi card

```
# ssh mic0 '/etc/init.d/ibmodules start'
Starting ofed layer ... Done
```

and loading the `rdma_ucm` module

```
# ssh mic0 'modprobe rdma_ucm'
```

all necessary (virtual) InfiniBand components (e. g. `scif0` interface and `rdma_cm` device) are set up on Xeon Phi.

APPENDIX C. ROUTING WITHIN THE XEON PHI CLUSTER

This section describes the setup of the flat network and routing for the communication across all OS instances, i. e. host nodes and Phi cards.

Communication between Xeon Phi card `X` and its host is over the network interface pair `micX`-$\texttt{mic0}_\texttt{X}$, where $\texttt{mic0}_\texttt{X}$ is the predefined network interface on card `X`. Both `micX` and $\texttt{mic0}_\texttt{X}$ spawn a separate subnetwork within the compute node. Communication between Xeon Phi cards within the same host requires to set up IP forwarding across the respective `micX` host interfaces.

Extending the setup to allow Xeon Phi cards in different compute nodes to communicate with each other is described hereafter for the nodes `bphi1` and `bphi2` of our Xeon Phi cluster.

1) Host interfaces: Both host nodes uses their interface `eth0` for communication over Gigabit Ethernet. We assign IPs `172.20.6.21/22` and `172.20.6.22/22` to them, and set MTU 9000 for both of the two.

2) Xeon Phi interfaces: On bphi1 and bphi2 we configure the host interface mic0 with IP 172.22.1.1/24 and 172.22.2.1/24 respectively. The Xeon Phi cards in bphi1 and bphi2 use their host's mic0 interface as gateway and have IPs 172.22.2.2/24 and 172.22.1.2/24 assigned to their own mic0 interface. Again, all interfaces are configured with MTU 9000. The respective changes can be applied to the mic0.conf file, for CentOS and MPSS 2.1 located in /etc/sysconfig/mic.

3) Routing: The private subnetwork 172.22.0.0/21 (the netmask value 21 follows from bphi3 and bphi4 assign IPs 172.22.3.1/24 and 172.22.4.1/24 to their mic0 host interfaces) is not known to HLRN-III. For a particular message from a sender within that subnetwork (any of the Xeon Phi cards), the network path thus has to be determined either by bphi1 or bphi2 itself. We therefore modified the routing for the network devices on bphi1 and bphi2 (file .../network-scripts/route.eth0, Fig. 9).

```
## file=.../route.eth0 on bphi1
GATEWAY0=172.20.6.22    # IP of bphi2
ADDRESS0=172.22.2.0     # Xeon Phi subnet
NETMASK0=255.255.255.0  # on bphi2

## file=.../route.eth0 on bphi2
GATEWAY0=172.20.6.21    # IP of bphi1
ADDRESS0=172.22.1.0     # Xeon Phi subnet
NETMASK0=255.255.255.0  # on bphi1
```

Figure 9. Network device configuration

Figure 10 shows a segment of bphi1's routing table. Routes to the Xeon Phis within bphi3 and bphi4 are also included. Messages send from bphi1's Xeon Phi to any other Xeon Phi use bphi[2,3,4] as intermediate stations along the respective network paths. The scheme can be easily expanded to more than 4 Xeon Phi nodes, but relatively fastly becomes cumbersome.

```
## Routing table of bphi1
Destination Gateway Genmask ......... Iface
........... ....... ....... ......... .....
172.22.4.0  bphi4   255.255.255.0     eth0
172.22.3.0  bphi3   255.255.255.0     eth0
172.22.2.0  bphi2   255.255.255.0     eth0
172.22.1.0  *       255.255.255.0     mic0
........... ....... ....... ......... .....
```

Figure 10. Segment of the routing table of bphi1. Messages from bphi1's Xeon Phi to any other Xeon Phi are routed first to the respective hosts, as intermediate stations along the network path, and then to the Xeon Phis.

4) IP Forwarding & NAT: For messages to be transferred to/from the Xeon Phi using IPv4 it is required that IPv4 forwarding on the host is enabled. On CentOS this can be done either by adapting the line net.ipv4.ip_forward=1 in /etc/ sysctl.conf, or by writing "1" into /proc/sys/net/ipv4/ip_forward. Further, forwarding rules have to be added to the IPtables file (/etc/sysconfig/iptables for CentOS). For our setup we allow forwarding to any of the Xeon Phis just if the sender belongs to the 172.22.0.0/21 subnetwork (Fig. 11).

```
## file=/etc/sysconfig/iptables on bphi1
.......................................
*filter
.......................................
-A FORWARD -i mic0 -s 172.22.0.0/21 -j ACCEPT
-A FORWARD -o mic0 -j ACCEPT
.......................................
COMMIT
*nat
.......................................
-POSTROUTING -o eth0 -j MASQUERADE
COMMIT
```

Figure 11. Segment of the IPtables file of bphi1. The filter-section contains forwarding rules for IPv4 messages to/from Xeon Phi. The nat-section is necessary for NFS mount on Xeon Phi (see below).

Since our Xeon Phi subnetwork is private, using HLRN-III services, e. g. NFS mount of the users' HOME directories on the Xeon Phi cards, requires to translate the Xeon Phis' network addresses into those known to HLRN-III. Our iptables file therefore contains Network Address Translation (NAT) entries (Fig. 11). All messages sent over the eth0 interface (connection to HLRN-III) inherit eth0's IP (achieved by -j MASQUERADE).

APPENDIX D. TMI CONFIGURATION

On both the host and Xeon Phi system a tmi.conf file need to created (default location: /etc) with the following content

```
## file=/etc/tmi.conf
psm 1.1 libtmip_psm.so " "
```

APPENDIX E. NFS MOUNTS ON PHI CARD

For NFS mount to work on Xeon Phi, for our setup the aforementioned NAT entries in /etc/sysconfig/iptables have been applied. In order to create the directories /home/b and /sw when (re)starting the MPSS service, the file INTEL_INSTALL_PATH/mic/filesystem/mic0.filelist must contain the following lines:

```
## file=INTEL_INSTALL_PATH/mic/filesystem/
##     mic0.filelist
...
dir /home/b 755 0 0
dir /sw 755 0 0
```

Similar to the host node, the Xeon Phi's `fstab` (located at `INTEL_INSTALL_PATH/mic/filesystem/mic0/ /etc/fstab`) needs to contain the NFS mount entries. It is particularly necessary to assign to all network interfaces along the network path used for NFS mount the same MTU (in our case MTU 9000).

Acknowledgment

The Intel Xeon Phi server and Intel TrueScale network components are kindly donated by Intel to ZIB within the activities of the "Research Center for Many-Core High-Performance Computing". We like to thank Michael Hebenstreit and Thorsten Schmidt from Intel for their help to optimize the IB network configuration. We further thank Dr. Michael Klemm from Intel for his support of performance optimization of workloads on Xeon Phi. Finally, we thank Thorsten Schütt for his optimization of BQCD on Xeon Phi.

References

[1] Finite-Volume Navier-Stokes procedure FreSCo. http://www.tuhh.de/alt/fds/research/current-projects/fresco.html.

[2] NEMO - Nucleus for European Modelling of the Ocean. http://www.nemo-ocean.eu.

[3] OpenFOAM. http://www.openfoam.org.

[4] TOP500. http://www.top500.org, November 2013.

[5] Torque: Intel Many-Integrated Cores (MIC) architecture configuration. Adaptive Computing, http://docs.adaptivecomputing.com/mwm/help.htm#topics/accelerators/mics.html, 2014.

[6] Torque: Prologue and epilogue scripts. Adaptive Computing, http://http://docs.adaptivecomputing.com/torque/Content/topics/12-appendices/prologueAndEpliogueScripts.htm, 2014.

[7] Sebastian Dressler and Thomas Steinke. Energy consumption of CUDA kernels with varying thread topology. *Computer Science - Research and Development*, pages 1 – 9, 2012.

[8] Werner Dubitzky, Assaf Schuster, Peter M. A. Sloot, Michael Schroeder, and Mathilde Romberg, editors. *Distributed, High-Performance and Grid Computing in Computational Biology , International Workshop, GCCB 2006, Eilat, Israel, January 21, 2007, Proceeding*, volume 4360 of *Lecture Notes in Computer Science*. Springer, 2007.

[9] Hebenstreit, M. *Configuring Intel Xeon Phi coprocessors inside a cluster*, January 2013.

[10] Heinecke, A. and Klemm, M. and Bungartz, H. J. From GPGPU to Many-Core: Nvidia Fermi and Intel Many Integrated Core Architecture. *Computing in Science and Engineering*, 14:78–83, 2012.

[11] Intel. Intel 12300 InfiniBand Switch Product Brief, March 2012.

[12] Intel. Intel TrueScale InfiniBand QLE7300 Series Product Brief, March 2012.

[13] Intel. Intel Xeon Phi Product Family Brief, Highly-Parallel Processing for Unparalleled Discovery, June 2013.

[14] Intel. *System Administration for the Intel Xeon Phi Coprocessor*, July 2013.

[15] Intel. *Intel Server System R2000GZ/GL Technical Product Specification - Rev 1.3.2*, February 2014.

[16] Jeffers, James and Reinders, James. *Intel Xeon Phi Coprocessor High Performance Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2013.

[17] Patrick May, Gunnar W. Klau, Markus Bauer, and Thomas Steinke. Accelerated microRNA-Precursor Detection Using the Smith-Waterman Algorithm on FPGAs. In Dubitzky et al. [8], pages 19–32.

[18] Nakamura, Yoshifumi and Stüben, Hinnerk. BQCD—Berlin quantum chromodynamics program. *PoS*, LATTICE2010:040, 2010.

[19] Nakamura, Yoshifumi and Stüben, Hinnerk. *BQCD Manual*, November 2011.

[20] S. Raasch and M. Schröter. PALM - a large-eddy simulation model performing on massively parallel computers. *Meteorol. Z.*, 10:363–372, 2001.

[21] Eric Stahlberg, Michael Babst, Daryl Popig, and Thomas Steinke. Molecular Dynamics with FPGAs: A Portable API Molecular Simulations with Hardware Accelerators: A Portable Interface Definition for FPGA Supported Acceleration. In *International Supercomputing Conference 2007 (ISC 2007)*, 2007.

[22] Eric Stahlberg, Daryl Popig, Debie Ryle, Michael Babst, Mohammed Anderson, and Thomas Steinke. Molecular Simulations with Hardware Accelerators. In *Reconfigurable Systems Summer Institute 2007 (RSSI 2007)*, 2007.

[23] Thomas Steinke, Kathrin Peter, and Sebastian Borchert. Efficiency Considerations of Cauchy Reed-Solomon Implementations on Accelerator and Multi-Core Platforms. In *Symposium on Application Accelerators in High Performance Computing (SAAHPC)*, 2010.

[24] Thomas Steinke, Alexander Reinefeld, and Thorsten Schütt. Experiences with High–Level Programming of FPGAs on Cray XD1. In *Cray Users Group (CUG 2006)*, May 2006.

[25] Joost VandeVondele, Matthias Krack, Fawzi Mohamed, Michele Parrinello, Thomas Chassaing, and Jürg Hutter. Quickstep: Fast and accurate density functional calculations using a mixed Gaussian and plane waves approach. *Computer Physics Communications*, 167(2):103 – 128, 2005.

[26] Q. Wang, S. Danilov, D. Sidorenko, R. Timmermann, C. Wekerle, X. Wang, T. Jung, and J. Schröter. The Finite Element Sea ice-Ocean Model (FESOM): formulation of an unstructured-mesh ocean general circulation model. *Geoscientific Model Development Discussions*, 6(3):3893–3976, 2013.

[27] Florian Wende, Frank Cordes, and Thomas Steinke. On Improving the Performance of Multi-threaded CUDA Applications with Concurrent Kernel Execution by Kernel Reordering. *Symposium on Application Accelerators in High-Performance Computing (SAAHPC)*, 0:74–83, 2012.

[28] Florian Wende and Thomas Steinke. Swendsen-Wang Multi-cluster Algorithm for the 2D/3D Ising Model on Xeon Phi and GPU. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 83:1–83:12, New York, NY, USA, 2013. ACM.