Journal of large-scale research facilities

Journal of large-scale research facilities, 8, A187 (2024) http://dx.doi.org/10.17815/jlsrf-8-187

Published: 26.04.2024

Noctua 2 Supercomputer

Paderborn Center for Parallel Computing (PC2) *

Instrument Scientists:
- Carsten Bauer, Paderborn University, email: carsten.bauer@uni-paderborn.de
- Tobias Kenter, Paderborn University, email: tobias.kenter@uni-paderborn.de
- Michael Lass, Paderborn University, email: michael.lass@uni-paderborn.de
- Lukas Mazur, Paderborn University, email: lukas.mazur@uni-paderborn.de
- Marius Meyer, Paderborn University, email: marius.meyer@uni-paderborn.de
- Holger Nitsche, Paderborn University, email: holger.nitsche@uni-paderborn.de
- Heinrich Riebler, Paderborn University, email: heinrich.riebler@uni-paderborn.de
- Robert Schade, Paderborn University, email: robert.schade@uni-paderborn.de
- Michael Schwarz, Paderborn University, email: michael.schwarz@uni-paderborn.de
- Nils Winnwa, Paderborn University, email: nils.winnwa@uni-paderborn.de
- Alex Wiens, Paderborn University, email: alex.wiens@uni-paderborn.de
- Xin Wu, Paderborn University, email: xin.wu@uni-paderborn.de

Management:
- Christian Plessl, Paderborn University, email: christian.plessl@uni-paderborn.de
- Jens Simon, Paderborn University, email: jens.simon@uni-paderborn.de

Abstract: Noctua 2 is a supercomputer operated at the Paderborn Center for Parallel Computing (PC2) at Paderborn University in Germany. Noctua 2 was inaugurated in 2022 and is an Atos BullSequana XH2000 system. It consists mainly of three node types: 1) CPU Compute nodes with AMD EPYC processors in different main memory configurations, 2) GPU nodes with NVIDIA A100 GPUs, and 3) FPGA nodes with Xilinx Alveo U280 and Intel Stratix 10 FPGA cards. While CPUs and GPUs are known off-the-shelf components in HPC systems, the operation of a large number of FPGA cards from different vendors and a dedicated FPGA-to-FPGA network are unique characteristics of Noctua 2. This paper describes in detail the overall setup of Noctua 2 and gives insights into the operation of the cluster from a hardware, software and facility perspective.

* **Cite article as:** Paderborn Center for Parallel Computing. (2024). Noctua 2 Supercomputer. *Journal of large-scale research facilities, 8*, A187. http://dx.doi.org/10.17815/jlsrf-8-187

1 Introduction

High-performance computing (HPC) is an integral part of to the modern technological landscape. HPC systems accelerate scientific research by providing the computing power needed to solve complex, large-scale problems through parallel processing on CPUs and customised, parallel processing with accelerators like GPUs and FPGAs. HPC systems foster collaborative research across disciplines by acting as shared resources within research institutions.

Figure 1: The Noctua 2 supercomputer operated at the Paderborn Center for Parallel Computing.

The Paderborn Center for Parallel Computing (PC2) is a national high-performance center in Germany and a scientific institute at Paderborn University. PC2 is member of the national HPC association NHR (*NHR Alliance*, 2023), the Gauß Alliance, the regional HPC.NRW competence network of North Rhine-Westphalia. Resource access is granted in a science-guided, competitive external peer-review process to researchers from universities in Germany. For the FPGA nodes, researchers from all over the world can also apply for resources.

The mission of PC2 is to advance interdisciplinary research in parallel computing and computational sciences with innovative computer systems. In cooperation with scientists and third party-funded research projects PC2 specializes in three areas: Atomistic Simulations, Optoelectronics and Quantum Photonics, and Machine Learning for Intelligent Systems. In each of these competence areas, PC2 possesses long-standing scientific expertise, which is demonstrated by competitive collaborative research projects, research infrastructures, high-profile personal projects and the prestigious awards the researchers have gained in these areas. Each competence area is created as interdisciplinary and cross-cutting in terms of scientific disciplines. The main contributions are to the fields of Physics, Chemistry and Engineering. The contributions also include the development of new methods for HPC simulation codes and contribution of these codes to widely used open source codes, e.g., CP2K (Kühne et al., 2020), i-Pi (Kapil et al., 2019), Quantum Espresso (Giannozzi et al., 2009) and numerical libraries like DBCSR (Borstnik et al., 2014). They are also driving forces for the development of libraries for emerging accelerator types (FPGAs, GPUs with Tensor Units) and computing paradigms, such as approximate and reconfigurable computing. The selected areas are also highly relevant and extend beyond Paderborn University. In particular, the methods and codes from atomistic simulations are very widely applicable for numerous scientific fields. Applications from the focus areas jointly generate an estimated demand of 25-50 % of the current global HPC work-

load. PC2 leverages the synergies of this special combination of services and research to satisfy the needs of computational sciences while addressing the big challenges in computing systems research: energy-efficiency, scalability and programmability.

Noctua 2 (see Figure 1) is the current flagship HPC system operated at PC2 and the main focus of this paper. Section 2 gives an overview of the overall components required to operate the system. This includes the actual nodes equipped with processors and accelerators, the network and storage subsystems and the software stack. In Section 3 the data center building is described to provide the infrastructure with power, cooling and protection systems. Next, in Section 4 the components and operation concepts of the FPGA partition are presented. The large number of FPGA nodes integrated into Noctua 2 and the dedicated FPGA-to-FPGA network are unique features that are part of the system. And finally, Section 5 concludes and summarizes the contents of the paper.

2 Noctua 2 HPC Cluster

In this section, the overall setup of Noctua 2 is described. Figure 2 provides an overview of the main components that are either part of the processing, storage and communication capabilities of the cluster or part of the infrastructure that is required to operate the system (mainly power and cooling).

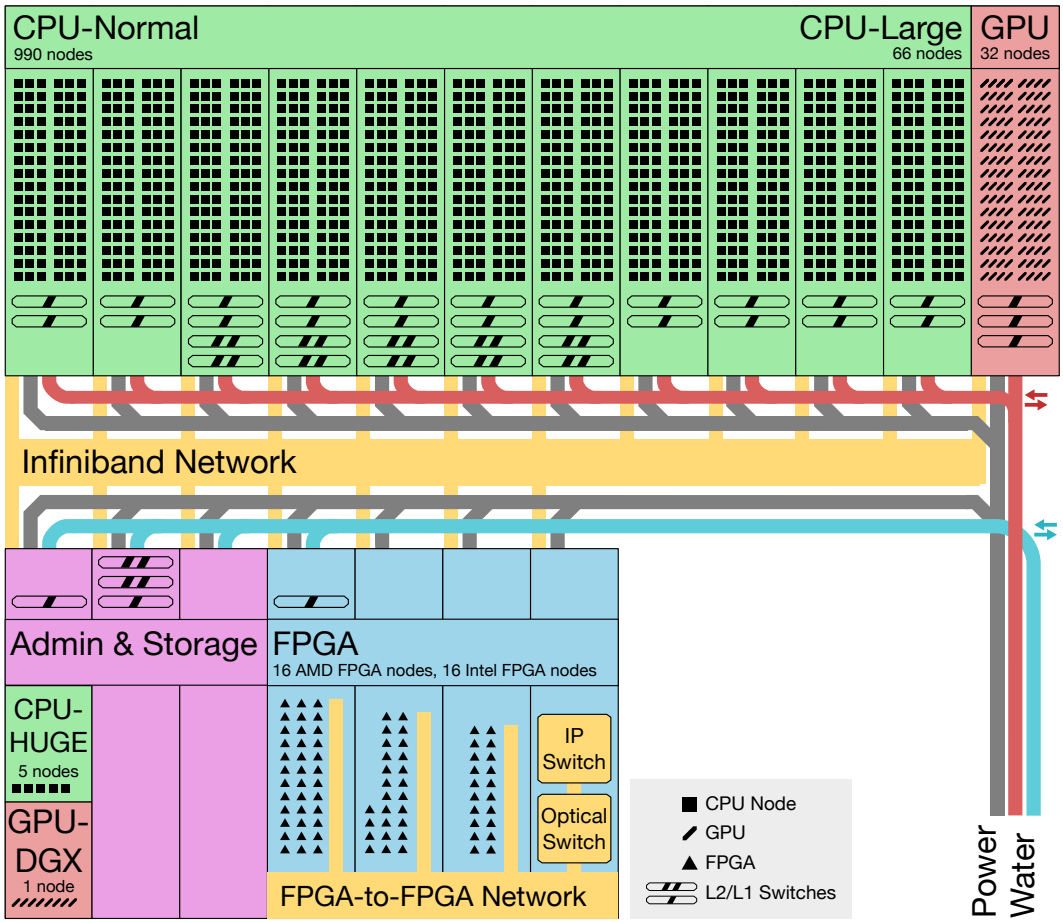

Figure 2: Schematic floor plan of Noctua 2 with racks containing nodes, network switches and accelerator cards together with the power and water cooling infrastructure.

2.1 Node Types and Quantities

Noctua 2 consists of compute nodes in different main memory configurations and nodes equipped with accelerator cards. The overall system has 1 126 nodes with a total of 144 128 CPU cores and 136 GPU and 80 FPGA accelerators. Table 1 lists the different node types and quantities.

Compute Nodes

Variant	Nodes	CPU*	Memory	Infiniband	Specialty
cpu-normal	990	EPYC 7763	256 GB	HDR 100	-
cpu-large	66		1 TB		
cpu-huge	5	EPYC 7713	2 TB		36 TB local SSD storage

GPU Nodes

Variant	Nodes	CPU*	Memory	Infiniband	Accelerator
gpu-a100	32	EPYC 7763	512 GB	2x HDR 200	4x Nvidia A100 40 GB
gpu-dgx	1	EPYC 7742	1 TB	HDR 200	8x Nvidia A100 40 GB

FPGA Nodes

Variant	Nodes	CPU*	Memory	Infiniband	Accelerator
fpga-amd	16	EPYC 7713	512 GB	HDR 100	3x Xilinx Alveo U280
fpga-intel	16				2x Bittware 520N

*All CPUs are dual-socket with 64 cores per socket.

Table 1: Overview of node types and quantities available in Noctua 2.

All compute nodes have a dual-socket CPU setup featuring two AMD EPYC CPUs, each with 64 cores. The 990 *cpu-normal* nodes constitute the largest fraction of compute nodes. The 66 *cpu-large* nodes contain 1 TB of main memory, while the five *cpu-huge* nodes provide 2 TB for applications demanding even more memory. Additionally, the *cpu-huge* nodes hold 36 TB SSD storage each for fast file access or transparent memory expansion. The 32 *gpu-a100* nodes give access to a total of 128 Nvidia A100 GPUs, each with 40 GB of HBM2 memory. The *gpu-dgx* node complements the system with eight Nvidia A100 GPUs to the GPU partition of the cluster. All cards of both GPU partitions are connected via SXM. Finally, the FPGA partition consists of 16 *fpga-amd* nodes with three Xilinx cards each and 16 *fpga-intel* nodes with two Bittware cards each. Details about the FPGA acceleration cards are described in Section 4.

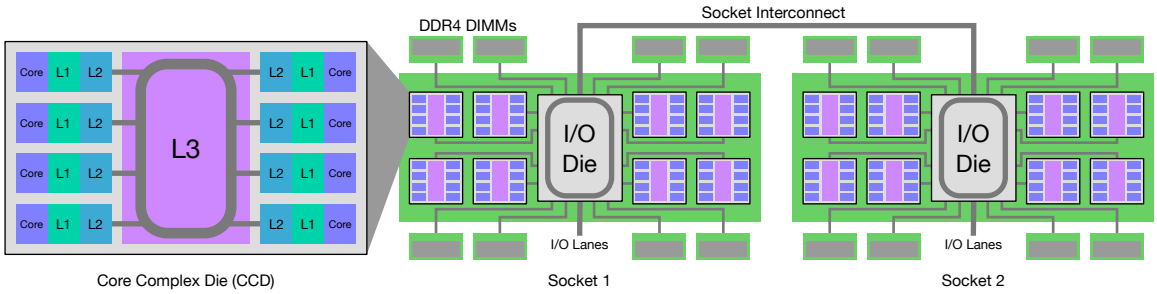

Figure 3: Topology of two AMD EPYC Milan CPU sockets.

2.2 CPU and Memory Hierarchy

With exception of the DGX node, all the compute nodes make use of AMD EPYC Milan data-center CPUs featuring AMD's Zen3 CPU core architecture. The EPYC CPUs make use of chiplet design to

accommodate multiple dies under one CPU package. The AMD 7763 and AMD 7713 CPU models used in Noctua 2 contain one I/O die that connects eight 8-core dies, main memory, I/O lanes and the second CPU socket as depicted in Figure 3. The 7763 CPU runs at 2.45 GHz base frequency and 3.5 GHz turbo clock at a TDP of 280 W. The *cpu-huge* and *fpga* nodes feature an AMD 7713 CPU with a lower TDP of 225 W and 2 GHz base clock. Each Zen3 core has access to 64 kB L1 and 512 kB L2 private CPU cache and 32 MB shared L3 per core-die. This makes a total of 4 MB L1, 32 MB L2 and 256 MB L3 cache per CPU (see Table 2). Finally, the Nvidia DGX provides a AMD EPYC 7742 Rome CPU with Zen2 core architecture, 2.25 GHz base clock and a TDP of 225 W. SMT is deactivated on all Noctua 2 compute nodes. Every CPU socket is connected via eight main memory channels to DDR4-3200 DIMMs.

EPYC 7763/7713 Caches

Type	Associativity	Size	Latency
L1 Instr	8-way set associative	32 kB per core	-
L1 Data			4-5 cycles (Integer), 7-8 cycles (FPU)
L2		512 kB per core	12 cycles
L3	16-way set associative	32 MB per 8 cores	46 cycles

EPYC 7742 Caches

Type	Associativity	Size	Latency
L1 Instr	8-way set associative	32 kB per core	-
L1 Data			4-5 cycles (Integer), 7-8 cycles (FPU)
L2		512 kB per core	12 cycles
L3	16-way set associative	16 MB per 4 cores	39 cycles

Table 2: Cache hierarchy of AMD EPYC processors (AMD, 2020).

The theoretical and measured performance of the CPU nodes is the following:

Double-precision FLOP/s @ 2.45 GHz

$$\begin{aligned}&\text{sockets} \times \text{freq.} \times \text{cores} \times \text{execution units} \times \text{elements per vector} \times \text{operations per element}\\ &= 2 \times 2.45\,\text{GHz} \times 64 \times 2 \times 4 \times 2\\ &= 5\,017.6\,\text{GFLOP/s}\end{aligned}$$

DDR4 RAM Data rate

$$\begin{aligned}&\text{sockets} \times \text{transfers} \times \text{bytes per channel} \times \text{channels per socket}\\ &= 2 \times 3.2\,\text{GT/s} \times 8\,\text{B/channel} \times 8\,\text{channels}\\ &= 409.6\,\text{GB/s}\end{aligned}$$

The actual CPU Test results with benchmark applications are summarized in Table 3.

2.3 GPUs

All the GPU nodes and the DGX node contain NVIDIA A100 SXM4 GPUs (*NVIDIA A100 Tensor Core GPU Architecture*, 2023), each of which has 40 GB HBM2 memory. The NVIDIA A100 GPU includes 54 Texture Processing Clusters (TPCs), consisting of 108 Streaming Multiprocessors (SMs) in total. Each SM contains 64 FP32 cores, 32 FP64 cores, and four tensor cores, thus resulting in 6912 FP32 cores, 3456 FP64 cores and 432 tensor cores per GPU. In addition, each SM has 256 kB register files and 192 kB combined L1 data cache and shared memory, which is configurable up to 164 kB. The

Metric	Variant	Performance
Peak FLOP/s LIKWID (Treibig et al., 2010)	1 core SP AVX-FMA	112.4 GFLOP/s
	1 core DP AVX-FMA	56.2 GFLOP/s
	128 cores SP AVX-FMA	10733.0 GFLOP/s
	128 cores DP AVX-FMA	5374.6 GFLOP/s
HPL (Petitet, 2004)	full node	4143.3 GFLOP/s
HPCG (Dongarra & Heroux, 2013)	full node	63.2 GFLOP/s
STREAM (McCalpin et al., 1995)	memory bandwidth, full node	370.3 GB/s

Table 3: Measurement results for selected CPU benchmarks.

40 MB L2 cache is shared among all SMs in an A100 GPU. The A100 GPU runs at 1.41 GHz boost clock frequency with the TDP of 400 W. The data transfer between the host CPUs and the A100 GPUs goes through the PCIe Gen4 interface, which provides 31.5 GB/s bandwidth per direction. In a GPU node, four A100 GPUs are interconnected via third-generation NVIDIA NVLink interconnects, providing up to 600 GB/s of direct GPU-to-GPU bandwidth. The DGX A100 node has additionally six second-generation NVIDIA NVSwitch fabrics that interconnect eight A100 GPUs using third-generation NVIDIA NVLink interconnects, which provide up to 600 GB/s of individual GPU-to-GPU bandwidth.

Theoretical Performance The theoretical performance of the GPU nodes[1] is the following:

CUDA-core FP64 FLOP/s @ 1.41 GHz

$$\begin{aligned}&\text{frequency} \times \text{FP64 cores} \times \text{operations per element}\\ &= 1.41\,\text{GHz} \times 3456 \times 2\\ &= 9.7\,\text{TFLOP/s}\end{aligned}$$

CUDA-core FP32 FLOP/s @ 1.41 GHz

$$\begin{aligned}&\text{frequency} \times \text{FP32 cores} \times \text{operations per element}\\ &= 1.41\,\text{GHz} \times 6912 \times 2\\ &= 19.5\,\text{TFLOP/s}\end{aligned}$$

Tensor-core FP64 FLOP/s @ 1.41 GHz

$$\begin{aligned}&\text{frequency} \times \text{streaming multiprocessors} \times \text{FP64 FMA} \times \text{operations per element}\\ &= 1.41\,\text{GHz} \times 108 \times 64 \times 2\\ &= 19.5\,\text{TFLOP/s}\end{aligned}$$

Tensor-core TF32 FLOP/s @ 1.41 GHz

$$\begin{aligned}&\text{frequency} \times \text{streaming multiprocessors} \times \text{TF32 FMA} \times \text{operations per element}\\ &= 1.41\,\text{GHz} \times 108 \times 512 \times 2\\ &= 155.9\,\text{TFLOP/s}\end{aligned}$$

Tensor-core FP16 FLOP/s @ 1.41 GHz

$$\begin{aligned}&\text{frequency} \times \text{streaming multiprocessors} \times \text{FP16 FMA} \times \text{operations per element}\\ &= 1.41\,\text{GHz} \times 108 \times 1024 \times 2\\ &= 311.9\,\text{TFLOP/s}\end{aligned}$$

[1]The performance for the tensor-cores are calculated without the fine-grained structured sparsity.

HBM2 Memory Data rate

$$\begin{aligned}&\text{HBM2 stacks} \times \text{bus width per stack} \times \text{pump rate} \times \text{frequency}\\ &= 5 \times 128\,\text{B} \times 2 \times 1.215\,\text{GHz}\\ &= 1\,555\,\text{GB/s}\end{aligned}$$

NVLink Bandwidth

$$\begin{aligned}&\text{bandwidth per link} \times \text{number of NVLink links}\\ &= 50\,\text{GB/s} \times 12\\ &= 600\,\text{GB/s}\end{aligned}$$

2.4 System Interconnect and Network Topology

All Noctua 2 nodes are connected to an Ethernet and an Infiniband network. The Infiniband network enables high-bandwidth, low-latency communication for compute nodes and the storage system within the compute cluster. 40 48-port Infiniband switches, supporting HDR 200 links, are used to build a fat-tree topology as depicted in Figure 4. Each compute node is connected via HDR 100 to one of the 28 level 1 switches. Two HDR 100 links are combined with a splitter and connected to one HDR 200 port. As an exception, the GPU nodes are using two HDR 200 links each. Every level 1 switch is connected to each of the 12 level 2 switches via HDR 200. The switches are distributed over the racks and reside in the top shelves. The used fat-tree topology achieves a 1:2 blocking factor. An evaluation of the interconnect measured between 1.5 and 2.0 µs in an MPI ping pong test depending on the chosen node pairs and due to cable length. The measured bidirectional bandwidth was roughly 24.6 GB/s.
An additional dedicated Ethernet network is used for administration and management tasks and connection to external networks.

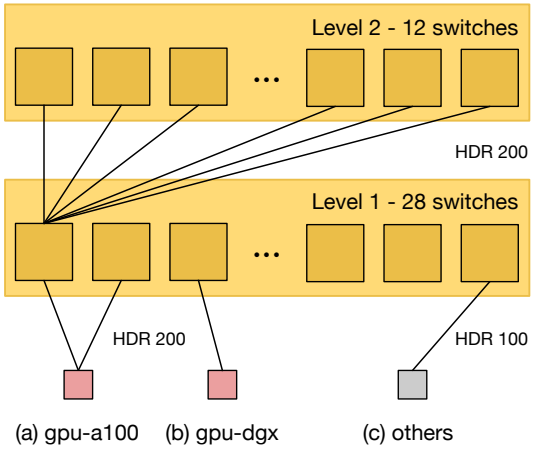

Figure 4: Infiniband network topology used in Noctua 2. Every level 1 switch is connected to all 12 level 2 switches. The nodes have different configurations: each *gpu-a100* node is connected via HDR 200 to two level 1 switches (a), the *gpu-dgx* node is connected to one level 1 switch via HDR 200 (b), while all other node types are connected via HDR 100 to one level 1 switch.

2.5 Storage Subsystem

Noctua 2 has a number of different file systems available for different purposes. Table 4 provides an overview and the key differences. Most of the file systems have quotas enabled. On *HOME* and

PC2DATA there are hard limits. If those are reached, no more data can be written. On the parallel file systems (*PC2PFS*), there are two limits *quota* and *limit*. Users can exceed the quota soft limit for a certain time (per default 14 days). After this time, no more data can be written. If users hit the hard limit, writing of further data is prohibited immediately. The limits are set for the storage capacity and number of files.

Name	Purpose	Permission Level		Backup
		Compute Nodes	Login Nodes	
HOME	User home directory for permanent, small data	read-write	read-write	yes
PC2DATA	Permanent project data for binaries, final results	read only		
PC2PFSN1	Parallel file system of Noctua 1			no
PC2PFS	Parallel file system of Noctua 2	read-write		
PC2DEPOT	Long term backup of research data	not available		yes

Table 4: Overview of the different storage types in Noctua 2.

The parallel file system is a Lustre File System with 6 PB capacity (DDN Exascaler 7990X with NVMe accelerator). The hardware consists of four servers with one expansion enclosure each, which makes a total of eight enclosures with a total of 658 HDD drives (12 TB each) and 28 SSD drives (6.4 TB each). The IO500 benchmark (*IO500*, 2023) was performed according to the SC21 specification and the results can be found in Table 5.

Metric	Performance	
ior-easy-write	68.5	GB/s
mdtest-easy-write	156.8	kIOPS
ior-hard-write	1.1	GB/s
mdtest-hard-write	94.4	kIOPS
find	1559.8	kIOPS
ior-easy-read	86.6	GB/s
mdtest-easy-stat	482.4	kIOPS
ior-hard-read	5.3	GB/s
mdtest-hard-stat	529.5	kIOPS
mdtest-easy-delete	107.6	kIOPS
mdtest-hard-read	160.9	kIOPS
mdtest-hard-delete	61.9	kIOPS

Table 5: Results of the IO500 benchmark performed on Noctua 2 according to the SC21 specification.

2.6 Software Stack, Services and System Management

Noctua 2 uses Red Hat Enterprise Linux as the operating system and Slurm (Yoo et al., 2003) as the job scheduling software. The users can use pre-installed software provided via Lua-based software environment modules (Lmod, McLay et al. (2011)). To increase the readability, the software packages are grouped into gateway modules (see Listing 1). Users can search for pre-installed software with a utility script (e.g. *find_module $NEEDLE*).

```
---------------- Gateway- and basic modules ---------------
   DefaultModules (L)    fpga    (*,G)    slurm/21.08.6
   all            (G)    lang    (G)      slurm/22.05.8-1 (L,D)
   bio            (G)    lib     (G)      system          (G)
   chem           (G)    math    (G)      toolchain       (G)
   compiler       (G)    mpi     (G)      tools           (G)
   data           (G)    numlib  (G)      vis             (G)
   debugger       (G)    pc2fs   (L)
   devel          (G)    perf    (G)
```

Listing 1: List of available gateway modules with pre-installed software (D: default module; G: gateway module; L: module is loaded *: module built for host, native FPGA and offload to FPGA)

Job Monitoring

To give users the opportunity to analyze performance issues of their jobs, a job-oriented monitoring framework called ClusterCockpit (Eitzinger et al., 2019) is provided. ClusterCockpit is integrated into Slurm and users can monitor their running and completed jobs in a web-based frontend. By clicking on the appropriate buttons, users can sort the list by different aspects (e.g. load on the CPUs, main memory used, memory bandwidth), display or hide metrics or apply filters on the list. As depcited in Figure 5, the metrics are plotted over time, which allows quick recognition of performance behavior. GPU jobs are also included in the job monitoring and users can inspect GPU metrics like compute utilization, memory utilization, and many more.

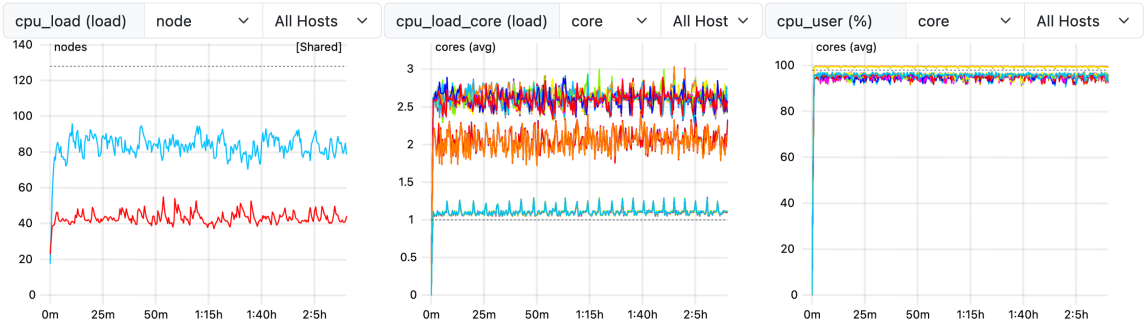

Figure 5: ClusterCockpit job monitoring gives users the opportunity to find problematic performance behavior. In this example, a CPU load imbalance between two nodes and a CPU core oversubscription can be recognized from the plotted CPU load metrics.

JupyterHub

Another service provided to the users is JupyterHub (*JupyterHub*, 2023). JupyterHub brings Jupyter notebooks to users and groups in an interactive computing environment. The hosted JupyterHub instance can not only spawn local notebooks on a dedicated machine, but the instance is also tightly integrated into Noctua 2 and Slurm. With pre-set environments users can directly spawn interactive jobs on any nodes of Noctua 2. Furthermore, all pre-installed software modules of Noctua 2 can also directly be loaded in the JupyterHub web interface.

System Management

Noctua 2 uses two admin nodes to manage the cluster and provide services like DHCP, DNS, NTP, NFS, Slurm, routing and IP-Forwarding, fabric management, monitoring and log aggregation. Pacemaker (*Pacemaker*, 2023) and Corosync (*Corosync*, 2023) are used for high-availability between the two nodes, an active/active concept is used, meaning that both head nodes are running a subset of services. Each node could run all services at once if one admin node should fail. Both nodes connect to an underlying storage appliance via SAS. Depending on the currently running services, several LVM volume groups get mounted to the node which is currently running the service and unmounted from the node which is not running the service. This way it is assured that no data corruption happens because two nodes cannot access the same data at the same time.

The whole cluster is configured with BlueBanquise (*BlueBanquise*, 2023) which is an Ansible (*Ansible*, 2023) based cluster manager. BlueBanquise uses plain-text files to generate configuration files for services, deploy software and, with helper scripts, provision node images. The configuration files are held in a local git repository, which is shared between the two admin nodes. Because almost every single line of configuration is generated from BlueBanquise, the whole cluster can be re-provisioned from scratch with just the BlueBanquise inventory, roles, and playbooks.

The node images are also created with the tools provided by BlueBanquise. The workflow is as follows: Create bare image containing only the operating system, mount bare image, roll-out the matching playbook. Then, Ansible parses the config and installs and configures the required programs and services directly into the mounted image. The image is packed, compressed and unmounted. Finally, a node can be selected to boot from the newly created image.

3 Data Center Building, Power and Cooling Infrastructure

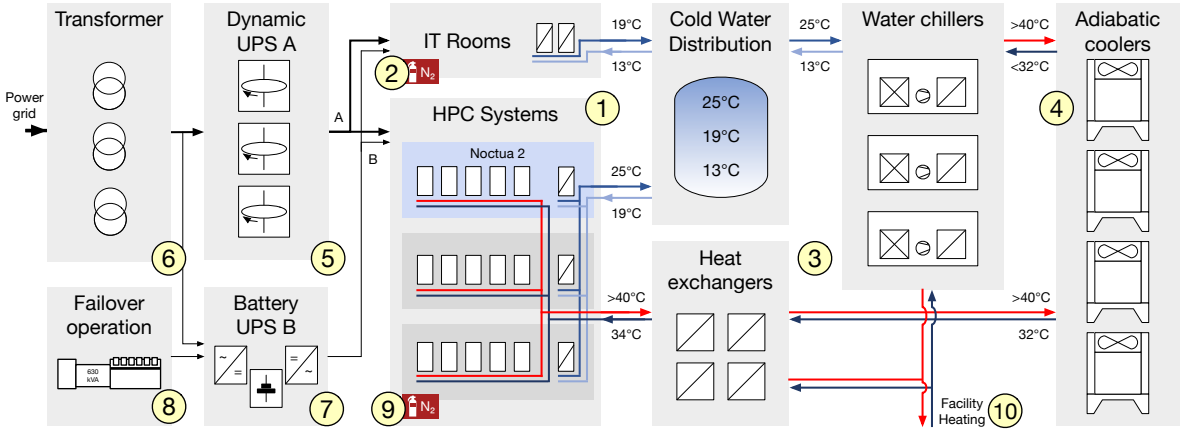

Figure 6: Schematic depiction of the power and water cooling infrastructure present in order to operate the Noctua 2 cluster.

Figure 6 shows the overall data center view, focusing on the technical infrastructure to run HPC systems. The HPC systems are located in the white space (1) with a floor area of over 330 m^2, 5 m freely usable ceiling height and 1 m deep raised floor. The white space is divided into three independent segments. Each segment can accommodate one large HPC system with its specific requirements for electricity and cooling. Currently, Noctua 2 occupies one segment and two segments are free. Additional rooms (2) accommodate network and storage infrastructure.

3.1 Power and Cooling

Power to the HPC systems is provided by a busbar system and ceiling-mounted outlet boxes, each with a maximum power capacity of over 1 MW per HPC system (see yellow rail in Figure 1). An electrical capacity of 2.6 MW is currently available for the HPC systems (6), and expansion to over 6 MW is in planning. The cooling systems of the data center are designed in such a way that the waste heat can be dissipated and reused in a highly efficient manner. To this end, the data center consistently relies on hot water cooling, which can dissipate at least 85 % of the waste heat. In order to be able to achieve a flow temperature of 32 °C for the hot water circuit all year round, the heat exchangers on the roof (3) can be sprinkled with water on particularly hot days. The humidification generates evaporative cooling and leads to a reduction of the temperature supplied by the heat exchanger, the so-called adiabatic cooling (4). The piping for the two cooling circuits, a cold water circuit with an inlet temperature of 19 °C, and a hot water circuit with an inlet temperature of >32 °C are located in the raised floor. The hot water loop can discharge up to 1 MW per segment, depending on the inlet and outlet temperatures allowed by the HPC system. The chilled water loop has a maximum total capacity of up to 700 kW and also supplies the chillers for air cooling (3). Due to the high temperature in the return of the hot water circuit, the waste heat can be used to heat buildings, for which a local heating network is being built on the campus of Paderborn University (10). Only a maximum of 15 % of the waste heat will be dissipated in the traditional way via air cooling, which is much more energy-intensive due to the generation of cold by compression chillers and distribution by fans.

3.2 Fault Tolerance and Protection Systems

Fail-safe operation requires specially designed power supplies for the HPC systems and the operation-critical technical systems. Therefore, two separate power supply lines A and B are always used for the core components of the HPC systems which include the data storage systems, server systems for administrative purposes and the core switches of the network interconnect. The concept is ensured by a power supply line A protected by a dynamic uninterruptible power supply (UPS) system (5) (line filter) and power supply line B protected by a battery-backed uninterruptible power supply (7). In the event of a prolonged failure of the mains power supply, the battery UPS is supplied with power via an emergency power system with a diesel engine (8). This setup is technically complex and rather maintenance-intensive due to the batteries used and not ideal from the sustainability perspective because of the increased energy consumption. To counteract this disadvantage, power supply line B is rated at 400 kW, the minimum size required for this function. Supply line A thus not only represents the second leg of supply for the most important IT components, but is also the only and thus central supply path for the large number of computing nodes. In the mains filter system, power is regenerated by a motor-generator combination, filtering out disturbances/fluctuations that may enter the building through the mains. A rotating flywheel in the system stores enough energy to bridge power outages for at least half a minute. Supply line A is designed for an output of over 2 MW (up to 6 MW is planned), so the high efficiency of the mains filter system benefits the entire data center.

Operation-critical technical systems such as pumps, fans of the air-circulation cooling units, chillers, etc. are only protected via the emergency failover operation system, since this is available after a short downtime of just a few seconds and the systems can easily tolerate this brief interruption. Furthermore, an early fire detection system and fire detection sensors provide for alarming and, if necessary, extinguishing fire by introducing nitrogen (9).

3.3 Efficiency of Operation

The efficient operation of HPC systems is becoming a more important criterion and is also required by the legislator in the current and future requirements for the sustainability of data centers. Noctua 2

features direct liquid cooling (DLC) for high coolant temperatures and thus makes optimum use of the efficiency potential of the indirect free cooling of the data center. Power Usage Effectiveness (PUE) is a key figure for estimating the energy efficiency of a data center. PUE is expressed as a ratio, determined by dividing the total energy entering a data center by the energy utilized to operate the IT equipment within the data center. The closer the value is to 1.0 the more energy-efficient the data center is and the better its energy balance. During the evaluation in the first month of operation, a PUE value of 1.1 was confirmed for Noctua 2.

4 FPGA Infrastructure

FPGA acceleration can provide performance gains and energy savings which are especially important in the HPC domain. Due to the ability to customize the accelerator hardware architecture to the algorithm at hand (hardware/software co-design), FPGAs are a promising technology which can be utilized to improve the energy efficiency of data centers by implementing architecture specialization. FPGAs are still expanding quickly in terms of performance, mostly through area scaling, and there are a variety of PCIe-based FPGA accelerator cards available with dedicated memory and network connectivity. However, unlike GPUs, FPGAs are still novel territory for HPC vendors, which up to now requires special expertise to specify and operate HPC systems with FPGAs. The PC2 offers the expertise to operate FPGAs, with a unique background in FPGA research and operation in the German HPC landscape and beyond. During recent years, multiple generations of state-of-the-art HPC systems, which are specifically tailored to FPGA acceleration have been, or are still, operated. On top of ever-increasing capacity of the FPGAs, these systems provided practical experience with different development flows, system level integration of the FPGAs and, more recently, also multi-FPGA operations.

In terms of publicly available HPC or cloud infrastructure with FPGAs, notable platforms and services include Amazon EC2 F1-instances (*Amazon EC2 F1-Instances*, 2023), Microsoft Azure (Collier & Shahan, 2015), Intel DevCloud (*Intel DevCloud*, 2023), and the AMD HACC (Heterogeneous Accelerated Cluster Computing) program (*AMD HACC Program*, 2023). Amazon EC2 F1-instances, offered by AWS, enable FPGA-accelerated compute resources for diverse applications. Microsoft Azure integrates FPGA acceleration, particularly in network and compute workloads, enhancing its cloud capabilities. Intel's DevCloud provides an accessible environment for FPGA development and experimentation with oneAPI, simplifying access to FPGA resources and tools. The AMD HACC program, combining FPGAs, GPUs and CPUs, showcasing the potential of heterogeneous computing. These platforms collectively offer versatile solutions for workload acceleration and high-performance computing across various domains.

This section starts with a general overview of the FPGA partition of Noctua 2. Then, the integration of the FPGA accelerators into the overall HPC system is described, details on the dedicated FPGA-to-FPGA network are given and finally the supported tool flows in order to enable our users to program the FPGAs are outlined.

4.1 Hardware Architecture and Design

A schematic view of the FPGA partition of Noctua 2 is depicted in Figure 7. It consists of 32 FPGA nodes with a total of 80 high-end FPGA cards from two vendors. All FPGA cards are physically connected to form a customizable direct FPGA-to-FPGA network with the help of an optical switch (Calient S320). In addition, an Ethernet switch (Huawei CE 9860) is also connected to the optical switch, in order to support packet-switched FPGA-to-FPGA communication. More details on the FPGA-to-FPGA networks are described later in Section 4.2.1. The 32 FPGA nodes are separated into

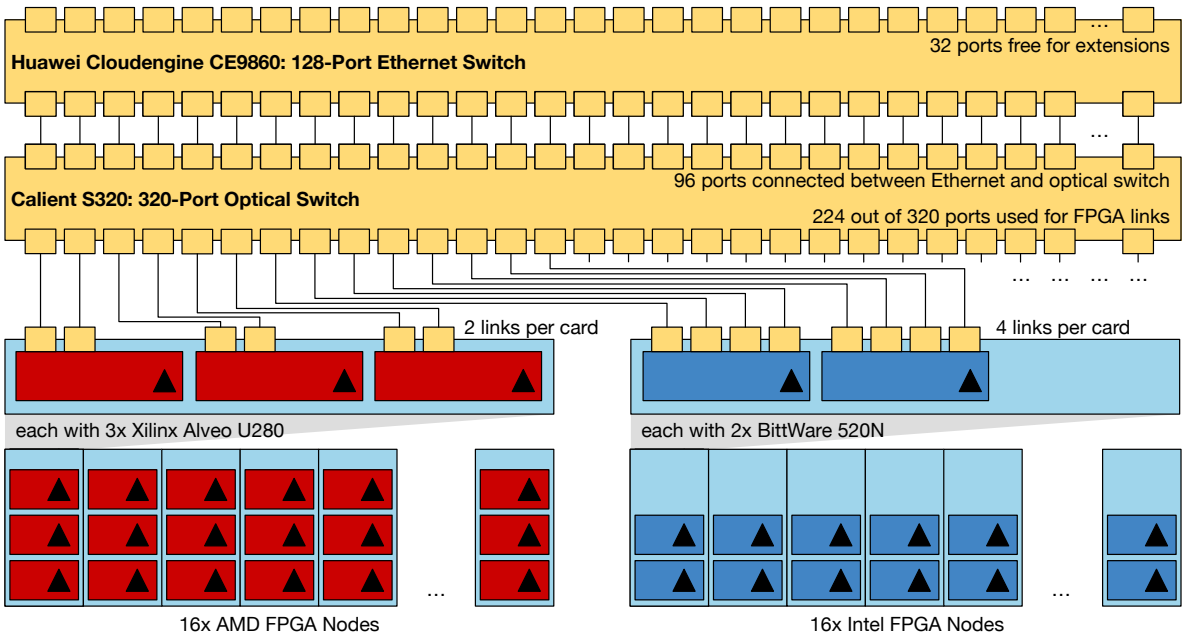

Figure 7: Schematic view of FPGA partition and FPGA-to-FPGA interconnect.

two partitions with 16 nodes each: 1) AMD FPGA nodes, each with three Xilinx U280 Alveo FPGA cards, and 2) Intel FPGA nodes, each with two BittWare 520N cards equipped with Intel Stratix 10 FPGAs.
There is also a custom FPGA partition consisting of four nodes. The custom FPGA nodes are used as a test bed for technology exploration of new FPGA cards in low volume. The FPGA cards included in this partition and the overall setup are changing and therefore this small subset is not further described.

The host system in all FPGA nodes is identical. It consists of two AMD EPYC Milan 7713, 2.0 GHz, each with 64 cores, 512 GB of main memory and 480 GB local SSD storage. This gives a total of 128 CPU cores per node, which is identical to the nodes in the main CPU partition. The processors of the FPGA nodes run at a slightly lower frequency compared to the CPU nodes to reduce the thermal power dissipation. With this configuration, the users can develop, compile and emulate their designs on any node of the cluster and only use the FPGA nodes for hardware execution. As described in Section 3, the FPGA nodes are part of the air-cooled racks of Noctua 2.

FPGA Accelerator Cards

Next, the hardware details and differences between the two main FPGA cards installed in Noctua 2 are discussed. Table 6 gives a comparison of the key features and outlines the main differences: In addition to 32 GB DDR on-board memory, each Xilinx U280 card has 8 GB of HBM2 (high bandwidth memory), which is not present on the BittWare cards. HBM can significantly improve the performance of computer systems, especially for memory-intensive tasks. Even though the cards have the same amount of on-board DDR memory (32 GB per card), the Xilinx U280 cards offer two memory interfaces with a total maximum bandwidth of 38.4 GB/s, while the BittWare cards divide the 32 GB over four banks of DDR memory with a total maximum bandwidth of 76.8 GB/s. Finally, each Xilinx U280 card has only two network interfaces for the FPGA-to-FPGA network, while the BittWare cards have four network interfaces per card.

	AMD FPGA Nodes	Intel FPGA Nodes
Cards Installation Year	2022	2018
Number of Nodes	16	16
Accelerator Cards		
Number of Cards	3x Xilinx Alveo U280	2x BittWare 520N
FPGA Type	UltraScale+ (XCU280)	Stratix 10 GX 2800
Lithography	16 nm	14 nm
Each Card		
Host Interface*	PCIe Gen3 x16	PCIe Gen3 x8
DDR Memory	2x 16 GB DDR4	4x 8 GB DDR4
DDR Bandwidth	38.4 GB/s	76.8 GB/s
High-Bandwidth Memory	8 GB HBM2	-
HBM Bandwidth	460 GB/s	-
Network Interfaces*	2x QSFP28 (100 Gbit/s)	4x QSFP+ (40 Gbit/s)
Thermal Design Power	225 W	225 W
Host Server		
CPUs	2x AMD Milan 7713, 2.0 GHz, each with 64 cores	
Main Memory	512 GB	
Storage	480 GB in local SSD. Rest in shared storage (see Section 2.5).	

*Correspond to the effectively usable values, implemented by the FPGA shell and used in our system.

Table 6: Comparison of the FPGA accelerator cards of Noctua 2.

4.2 System Integration

This sections gives details about the integration of the FPGA accelerator cards on system level, into the workload manager and on the dedicated FPGA-to-FPGA network.

4.2.1 On Demand Version Configuration of FPGA Nodes

Based on the insights gained from prior FPGA systems, three essential requirements for supporting FPGA development flows in a production HPC system have been identified:

1. Software Tools: Provide pre-installed software tools and regular updates.
2. Synthesis Infrastructure: Provide compilation infrastructure for FPGA circuit implementation tools.
3. Keep Compatibility: Provide ability to run designs created with previous tool versions.

1. Software Tools The programming tool chains are still progressing on both ends, the high-level synthesis (HLS) step which translates the high-level code (typically C++-based) to a design in hardware description language (HDL) and the backend flow which maps the HDL design to the physical FPGA resources. The HLS step has been significantly evolving with new features and optimizations over the last years and will continue to do so. It is therefore of paramount importance that regular updates are received and can thereby made available to all users. In order to provide the FPGA software tools, a dedicated Lmod gateway (see general Lmod description in Section 2.6) has been created to group all FPGA related software tools, drivers and utilities in one collection. The FPGA gateway structure is outlined in Listing 2.

For the development of FPGA designs, the focus is strongly on high-level synthesis tool flows, which have further developed over the last years. For the Xilinx Alveo U280 boards, the Xilinx Vitis HLS and Vivado tool flows are provided to build accelerator designs using high-level descriptions written in C++. Additionally, Xilinx published the Vitis Libraries, an extensive collection of accelerator building blocks covering linear algebra, signal processing and data processing applications, targeting the Alveo boards. This availability of libraries further lowers the barrier to the use of FPGAs. Meanwhile, Intel uses SYCL as the main high-level design language for FPGAs with compiler support being integrated in Intel oneAPI. The host application and the accelerator can be programmed in a single source code written in Data Parallel C++ (DPC++), a slightly extended variant of SYCL. This development flow, along with the classic OpenCL-based tool flows, is available for all our BittWare 520N FPGA boards on Noctua 2.

```
$ module load fpga              # Load FPGA gateway module
$ module available              # Show available modules.

List of the supported base bitstream or shell versions for Intel and Xilinx cards (see Section 4.2.1).
bittware/520n/20.4.0_hpc    (D) # (D) = default version.
bittware/520n/20.4.0_max
[...]
xilinx/xrt/2.13
xilinx/xrt/2.14             (D)

List of the supported development tool flows for Intel and Xilinx cards (see Section 4.2.1).
intel/oneapi/23.1.0         (D)
intel/opencl_sdk/21.4.0     (D)
[...]
xilinx/vitis/23.1           (D)
xilinx/vivado/23.1          (D)

List of FPGA utilities, for example to use direct FPGA-to-FPGA network (see Section 4.2.2).
intel/testFPGAlinks
intel/channel_emulation_patch
changeFPGAlinks
```

Listing 2: FPGA gateway module to provide software tools and utilities.

2. Synthesis Infrastructure The FPGA synthesis (translating HLS code into the configuration for the FPGAs) is a very complex and time consuming process that involves many optimization steps. It is often necessary to explore multiple synthesis options and iterate on the design to find the best performing design or balance between trade-offs like performance, area utilization, and power consumption. This is a CPU and memory intensive task. All nodes of Noctua 2 provide the required software infrastructure to perform these synthesis jobs, such that it is not necessary to block specialized accelerator nodes with them. The quality-of-service feature of Slurm is used to give a limited number of FPGA synthesis jobs a higher priority. Users can use the Slurm command *#SBATCH -q fpgasynthesis* to give their FPGA bitstream synthesis jobs a higher priority for at most 10 synthesis jobs per user.

3. Keep Compatibility Generally, C++-based designs are source-compatible with new tool versions, but since it takes many hours to compile a bitstream with a specific software tool and bitstream version, the capability to reuse existing bitstreams beyond the tool update cycles is highly desirable. This capability is also important for reproducibility of results. To use a bitstream, which was created with a specific tool version, the FPGAs need to be configured with a matching firmware, a so-called base bitstream or shell, see Figure 8. This base bitstream implements the communication with the

host CPU via PCIe, provides access to the memory on the FPGA card, and enables quick configuration for the accelerator bitstreams by utilizing partial reconfiguration at runtime. Finally, the software drivers matching the tool version need to be loaded after the base bitstream is configured.
This functionality is integrated with the regular Slurm workload manager. With a *--constraint* argument, see example in Listing 3, the user specifies, which base bitstream and tool version the allocated node or nodes should provide. These versions need to match the versions used in the synthesis process. The workload manager will initially try to allocate nodes, which already provide the requested configuration. If this is not possible, then other nodes will be allocated and, transparently to the requesting user, will be configured with the required base bitstream, rebooted to bring up the PCIe connection with the correct settings, and have the matching drivers installed. Therefore, after a few minutes, the allocation request can be served with freshly configured nodes. Rebooting whole nodes for specific user requests is safe, because FPGA nodes are provisioned only in exclusive mode (no node sharing between users). This also ensures the isolation of FPGA jobs from a security perspective. This is realized using a customized variant of the Slurm *node features* plugin (*Slurm Node Features Plugin Programmer Guide*, 2023) and custom scripts that reconfigure the nodes and reprogram the FPGAs according to the user's specification. The FPGA Lmod gateway in Listing 2 shows some of the different variants of base bitstreams or shells that can be used as a constraint argument.

```
By requesting a node in the FPGA partition with the constraint bittware_520n_20.4.0_max,
the base bitstream and driver for the BittWare card in version 20.4.0 will be provided for this job.
srun --partition=fpga --constraint=bittware_520n_20.4.0_max ./fpga_appl
```

Listing 3: Allocating FPGA Node with specific base bitstream.

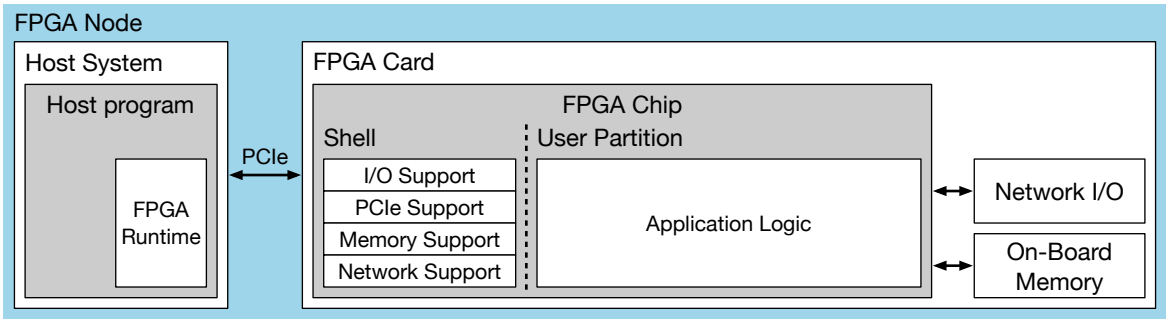

Figure 8: Schematic structure of an FPGA node with one FPGA card. The programmable area of the FPGA chip is separated into shell and user partition.

4.2.2 Dedicated FPGA Interconnect Infrastructure

With FPGAs arriving in the HPC domain, scaling of FPGA accelerated codes will become just as important as their contributions to single node performance and efficiency. The conventional approach to integrate accelerators like GPUs into HPC systems is to rely on the host and its networking infrastructure to communicate with other nodes and accelerators. The Noctua 2 FPGA nodes fully support that approach using InfiniBand as high-speed network technology which integrates them uniformly with the rest of the Noctua 2 system (see Section 2.4). However, FPGAs are also well suited for a more direct integration into high-speed networks. As already outlined in Section 4.1, the BittWare 520N cards provide four QSFP+ and the Xilinx Alveo U280 cards provide two QSFP28 network ports. In Noctua 2, all FPGA network ports are equipped with optical transceivers. When connected via electrical or optical pluggable transceivers, the ports create a direct point-to-point link

with up to 100 Gbit/s (Intel 40 Gbit/s, Xilinx 100 Gbit/s). BittWare BSPs with the *_max* suffix have direct FPGA-to-FPGA network support included in the BSP. The BSPs with the *_hpc* do not include the networking support in order to save resources. For the Xilinx cards, the FPGA-to-FPGA network support is not included in the shell, but can be included into the user logic, depending on the user needs.

A highly flexible solution to support arbitrary point-to-point links was added with the installation of a Calient S320 Optical Circuit Switch (see Figure 7). All FPGA ports are physically connected to the optical switch. Out of the 320 available ports of the optical switch, 224 are used for the FPGA links:

$$\begin{aligned}&(\text{number of AMD nodes} \times \text{number of cards per node} \times \text{number of links per card}) + \\ &(\text{number of Intel nodes} \times \text{number of cards per node} \times \text{number of links per card}) \\ &= (16 \times 3 \times 2) + (16 \times 2 \times 4) = 224\end{aligned}$$

The remaining 96 ports of the optical switch are connected to the Huawei CE 9860 Ethernet switch.

4.2.3 Dedicated FPGA Interconnect System Integration

Along with a job request, a user can request any interconnect topology for FPGA-to-FPGA connections and the workload manager will, along with the node allocation, establish the requested connections through the optical switch. Listing 4 shows an example to request a pairwise connection for the FPGA interconnect for one FPGA node. The FPGA application can use all available serial channels of one FPGA card to communicate to the respective channels of the other FPGA card in the same node, see Figure 9a as an example for a Intel FPGA node featuring two FPGA cards with four channels each.

```
srun --partition=fpga [...] -N 1 --fpgalink=pair ./fpga_appl
```

Listing 4: Examples for pairwise connection with one node. See visualization in Figure 9a.

In addition to predefined topologies (*pair* in previous example), a job request can also specify every possible individual connection between any links of the allocated FPGA cards. The following request in Listing 5 with individual links specified results in the same topology as the pair example in the previous Listing 4.

```
srun --partition=fpga [...] -N 1
  --fpgalink=n0:fpga0:ch2-n0:fpga1:ch2 --fpgalink=n0:fpga0:ch0-n0:fpga1:ch0
  --fpgalink=n0:fpga0:ch1-n0:fpga1:ch1 --fpgalink=n0:fpga0:ch3-n0:fpga1:ch3
  ./fpga_appl
```

Listing 5: Same pairwise connection with individual links specified. See visualization in Figure 9a.

Allocations with direct FPGA-to-FPGA connections can also target multiple nodes, as shown in Listing 6 with two nodes forming a *clique* topology, see Figure 9b for the visualization.

```
srun --partition=fpga [...] -N 2 --fpgalink=clique ./fpga_appl
```

Listing 6: Examples for clique topology with two node. See visualization in Figure 9b

The topology can also be altered during an allocation with a custom utility called *changeFPGAlinks*. The following example in Listing 7 changes the previously allocated *clique* topology during runtime into a *pair* topology with two nodes, see Figure 9c.

```
changeFPGAlinks
  --fpgalink=n1:fpga0:ch2-n1:fpga1:ch2 --fpgalink=n1:fpga0:ch0-n1:fpga1:ch0
  --fpgalink=n0:fpga0:ch2-n0:fpga1:ch2 --fpgalink=n0:fpga0:ch0-n0:fpga1:ch0
  --fpgalink=n0:fpga0:ch1-n0:fpga1:ch1 --fpgalink=n0:fpga0:ch3-n0:fpga1:ch3
  --fpgalink=n1:fpga0:ch1-n1:fpga1:ch1 --fpgalink=n1:fpga0:ch3-n1:fpga1:ch3
```

Listing 7: Example to change the topology at runtime. See visualization in Figure 9c

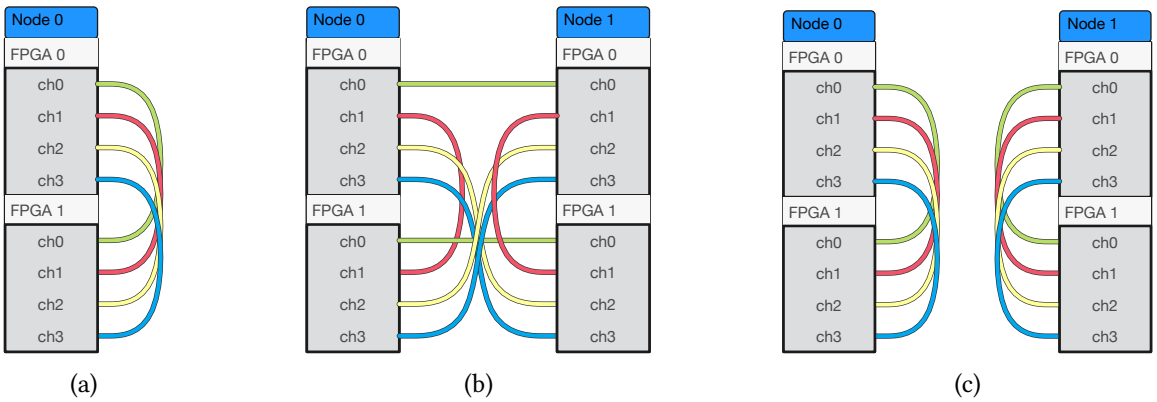

Figure 9: FPGA-Link GUI Examples: (9a) Single FPGA node with fully paired FPGAs, (9b) Two FPGA nodes with clique topology. (9c) Two FPGA nodes, each fully paired.

In order to guide the user to generate the desired configuration, a web GUI (*FPGALink-GUI*, 2023) is provided. The visualization depicted in Figure 9 are generated by the GUI.

4.2.4 Types of Dedicated FPGA Interconnects

From the perspective of the FPGA application logic, the point-to-point links can be used in two configurations:

1. Circuit-Switched: Simple serial transmission, which serves as a protocol-agnostic and transparent connection layer.
2. Packet-Switched: Flexible transmission with packets and routing.

1. Circuit-Switched FPGA-to-FPGA Network The simple serial transmission through these channels suits the pipelining and data streaming concepts for FPGAs very well and is even frequently used inside single-FPGA designs to exploit task-level parallelism. Hence, with a suitable topology of such serial links, applications can scale over multiple FPGAs without ever requiring latency-intensive communication via the hosts and without additional data introduced by the network protocols and mechanisms to manage and deliver packets effectively. The HPC Challenge for FPGA (Meyer et al., 2020) contains optimized benchmark implementations for this kind of network infrastructure which were used in Meyer et al. (2023) to evaluate the latency and bandwidth advantages compared to the CPU-centric communication via the host.
De Matteis et al. (2019) have presented a programming abstraction for streaming interfaces that includes multi-hop routing for this infrastructure when it was first available in Noctua 1. A similar concept was realized on the Cygnus system (Boku et al., 2023) at University of Tsukuba, where 64

Stratix 10 FPGAs physically connected in a fixed double torus topology use the CIRCUS (Fujita et al., 2020; Kikuchi et al., 2023) infrastructure for routing and collective communications. While for individual point-to-point connections, the performance impact of routing layers is mostly just increased latency, multiple point-to-point connections using several hops will directly compete for bandwidth on some routes compared to the circuit-switched direct connections possible with Noctua 2.

2. Packet-Switched FPGA-to-FPGA Network The technical infrastructure introduced to establish point-to-point connections between FPGA ports can also be used to connect the FPGA ports to the ports of the Ethernet switch (see Figure 7). With this setup, up to 96 FPGAs can be used to form a single-hop packet-switched network. The Ethernet switch is configured to create separate Virtual Local Area Networks (VLANs) for every compute job to prevent the failure of a job because of misconfigured concurrent jobs. The Ethernet switch is fully integrated into the overall FPGA interconnect infrastructure. The desired topology with Ethernet links can be configured with the *changeFPGAlinks* utility as shown in Listing 8 or with the web GUI (*FPGALink-GUI*, 2023) as depicted in Figure 10.

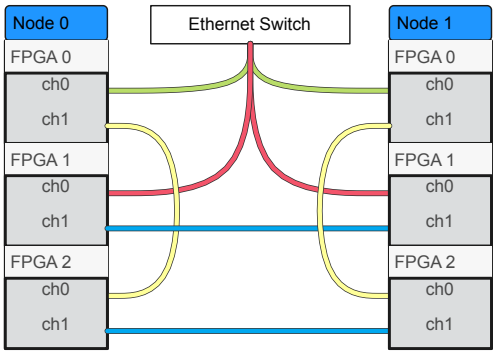

Figure 10: FPGA-Link example including the Ethernet switch: The first channel of the first two FPGAs are connected to the Ethernet switch.

changeFPGAlinks
```
--fpgalink=n1:fpga0:ch1-n1:fpga2:ch0 --fpgalink=n1:fpga2:ch1-n0:fpga2:ch1
--fpgalink=n1:fpga1:ch1-n0:fpga1:ch1 --fpgalink=n0:fpga2:ch0-n0:fpga0:ch1
--fpgalink=n0:fpga1:ch0-eth --fpgalink=n0:fpga0:ch0-eth
--fpgalink=n1:fpga0:ch0-eth --fpgalink=n1:fpga1:ch0-eth
```

Listing 8: Example FPGA-Link parameters for topology including the ethernet switch. See visualization in Figure 10

The shells of the Xilinx FPGAs pass the raw connections to the communication ports to the user logic. Therefore, users have to provide their own logic for the network protocols or use existing Intellectual Property cores with the desired functionality. There exist several open source projects which provide implementations of network stacks, such as UDP (*XUP Vitis Network Example (VNx)*, 2023) or TCP (He, Korolija, & Alonso, 2021). They abstract away most of the complexity of manually setting up and maintaining connections over a packet-switched network. An even higher abstraction layer offers the collective-communication library ACCL (He, Parravicini, et al., 2021). It provides a message-passing interface similar to MPI to the user application independent of the underlying network stack.
In ESSPER (Sano et al., 2023), an 8-node and 16-FPGA extension to the Fugaku supercomputer (Sato et al., 2020), the FPGAs are also connected with two 100Gbit/s ports each to a dedicated Ethernet

switch network. With their Virtual Circuit-Switching Network infrastructure, Ueno & Sano (2023) represent the physical Ethernet network to the FPGA user kernels as direct streaming connections like those that we realize physically with the circuit-switched network. Intel had presented a similar concept as Inter-Kernel Links (Balle et al., 2020). Unfortunately, as both implementations require support with different customized BSPs that are not available for the Bittware 520N cards, neither of these approaches is deployed on Noctua 2 for direct comparisons.

4.3 Applications using the FPGA Partition

As FPGA accelerators are getting deployed in quantity in HPC systems like Noctua 2, their performance and energy efficiency potential must also be brought to the relevant applications from the HPC domain. One approach that industry and academia use to tackle that topic is with libraries of different specialization degree, e.g. for dense (De Matteis et al., 2020; Gorlani et al., 2019; Hao et al., 2023; *Xilinx Vitis Libraries*, 2023) and sparse linear algebra (Jain et al., 2023; Song et al., 2022; *Xilinx Vitis Libraries*, 2023), and corresponding solvers (Meyer et al., 2022; Song et al., 2023; *Xilinx Vitis Libraries*, 2023; Zeni et al., 2021). When corresponding library functions on FPGA cards are invoked from multiple ranks of an MPI-parallel application, such libraries provide a direct path towards multi-FPGA applications. However, individual linear algebra operations that on CPUs or GPUs can often reach close to their respective peak performance are not necessarily the domain where FPGAs can provide most benefits. Also, bandwidth and latency limitations of data transfers between host and accelerator via PCIe can limit or even negate the advantages of this approach (Ramaswami et al., 2021).

Consequently, further customization of FPGA designs to more specific application needs, or usage of the dedicated FPGA-to-FPGA networks presented in Sections 4.2.2–4.2.4 are advisable. For the calculation of Electron Repulsion Integrals (ERIs), FPGAs can outperform (Wu et al., 2023) latest HPC CPUs at much improved energy efficiency by tailoring local memory layout and pipeline parallelism of different FPGA designs specifically to the angular momenta of different input types. In this context, a system with many FPGAs allows to distribute the workload to multiple heterogeneously configured devices in order to avoid overheads of frequent reconfiguration. ERI calculation is an important building block of many atomistic simulations that make up a large part of the Noctua 2 workload and we are working on further integration of the FPGA designs with production codes.

Another project where many FPGAs were employed for multiple month was the calculation of the 9^{th} Dedekind Number (Van Hirtum et al., 2023). Customization of bit level operations in a small graph with 128 nodes and a fine granular control of operation sequences of variable length along with suitable load balancing allowed each FPGA to outperform a 64 core CPU by around three orders of magnitude. Another project (Opdenhövel et al., 2023) that leverages bit level operations on FPGAs is from the bioinformatics domain. Here the speedups are more modest, because the more regular operations are better suited for the CPU code path, but the FPGAs still impress with their power efficiency.

The circuit-switched FPGA-to-FPGA network was leveraged for shallow water simulations (Faj et al., 2023) in an extension to an earlier single FPGA design (Kenter et al., 2021), also refer to Alt et al. (2023) on recent tooling progress. On top of a strong baseline performance, the tight integration of the streaming communication into the computation pipeline along with the customization of the FPGA interconnect topology to the spatial decomposition of the simulation domain enables good strong and weak scaling particularly for small to medium size problems, where CPUs and GPUs struggle to achieve good utilization. A similar observation has been made for N-body simulations with direct FPGA-to-FPGA communication (Menzel et al., 2021). Meanwhile, independent of each other, Stewart et al. (2021) and Sheng et al. (2023) have commercialized MD-simulations for drug discovery on their own directly interconnected multi-FPGA systems. Supercapacitor simulations (Prouveur et al., 2023) based on the N-body method have been ported to the Noctua 2 FPGA partition and perform best when

individual FPGAs are completely customized to one out of three different computation kernels, with load balancing performed by allocating more devices to the more time consuming kernels. At the given performance points, this workload still scales well when communicating via MPI on the host, and Contini et al. (2023) have been working to optimize efficiency for this type of communication scheme on the Alveo FPGAs in Noctua 2.

5 Conclusion

In conclusion, the paper provides a comprehensive overview of the Noctua 2 supercomputer. Inaugurated in 2022, Noctua 2 stands out as a HPC system, comprised of three distinct node types: CPU Compute nodes, GPU nodes, and FPGA nodes. The integration of diverse FPGA cards from different vendors, along with a dedicated FPGA-to-FPGA network, sets Noctua 2 apart from conventional HPC systems. The paper highlights the overall setup and operation of the cluster, offering valuable insights into its hardware, software, and facility aspects.

References

Alt, C., Kenter, T., Faghih-Naini, S., Faj, J., Opdenhövel, J.-O., Plessl, C., … Köstler, H. (2023). Shallow water DG simulations on FPGAs: Design and comparison of a novel code generation pipeline. In *Proc. Int. Conf. on High Performance Computing (ISC High Performance)* (pp. 86–105). Springer. Retrieved from https://doi.org/10.1007/978-3-031-32041-5_5 http://dx.doi.org/10.1007/978-3-031-32041-5_5

Amazon EC2 F1-Instances. (2023). [Online] https://aws.amazon.com/de/ec2/instance-types/f1/. (Accessed: 2023-10-04)

AMD. (2020). *Software Optimization Guide for AMD Family 19h Processors (PUB).*

AMD HACC Program. (2023). [Online] https://www.amd-haccs.io/. (Accessed: 2023-10-04)

Ansible. (2023). [Online] https://www.ansible.com. (Accessed: 2023-10-04)

Balle, S. M. ., Tetreault, M., & Dicecco, R. (2020). *Inter-kernel links for direct inter-fpga communication.* https://www.intel.com.br/content/dam/www/programmable/us/en/others/literature/wp/wp-01305-inter-kernel-links-for-direct-inter-fpga-communication.pdf.

BlueBanquise. (2023). [Online] https://bluebanquise.com. (Accessed: 2023-10-04)

Boku, T., Fujita, N., Kobayashi, R., & Tatebe, O. (2023). Cygnus - world first multihybrid accelerated cluster with GPU and FPGA coupling. In *Proc. Int. Conf. on Parallel Processing (ICPP) Workshops.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3547276.3548629 http://dx.doi.org/10.1145/3547276.3548629

Borstnik, U., VandeVondele, J., Weber, V., & Hutter, J. (2014). Sparse Matrix Multiplication: The Distributed Block-Compressed Sparse Row Library. *Parallel Computing*, *40*(5-6).

Collier, M., & Shahan, R. (2015). *Microsoft azure essentials-fundamentals of azure.* Microsoft Press.

Contini, N., Ramesh, B., Kandadi Suresh, K., Tran, T., Michalowicz, B., Abduljabbar, M., … Panda, D. (2023). Enabling Reconfigurable HPC through MPI-Based Inter-FPGA Communication. In *Proceedings of the 37th International Conference on Supercomputing* (pp. 477–487). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3577193.3593720 http://dx.doi.org/10.1145/3577193.3593720

Corosync. (2023). [Online] http://corosync.github.io/corosync/. (Accessed: 2023-10-04)

De Matteis, T., de Fine Licht, J., Beránek, J., & Hoefler, T. (2019). Streaming message interface: High-performance distributed memory programming on reconfigurable hardware. In *Proc. Int. Conf. on High Performance Computing, Networking, Storage and Analysis (SC).* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3295500.3356201 http://dx.doi.org/10.1145/3295500.3356201

De Matteis, T., de Fine Licht, J., & Hoefler, T. (2020). FBLAS: Streaming linear algebra on FPGA. In *Proc. Int. Conf. on High Performance Computing, Networking, Storage and Analysis (SC).* IEEE Press.

Dongarra, J., & Heroux, M. A. (2013). Toward a new metric for ranking high performance computing systems. *Sandia Report, SAND2013-4744*, *312*, 150.

Eitzinger, J., Gruber, T., Afzal, A., Zeiser, T., & Wellein, G. (2019). Clustercockpit—a web application for job-specific performance monitoring. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)* (pp. 1–7).

Faj, J., Plessl, C., Kenter, T., Faghih-Naini, S., & Aizinger, V. (2023). Scalable multi-FPGA design of a discontinuous galerkin shallow-water model on unstructured meshes. In *Proc. Platform for Advanced Scientific Computing Conf. (PASC)* (pp. 1–12). Retrieved from https://doi.org/10.1145/3592979.3593407 http://dx.doi.org/10.1145/3592979.3593407

FPGALink-GUI. (2023). [Online] https://pc2.github.io/fpgalink-gui. (Accessed: 2023-10-04)

Fujita, N., Kobayashi, R., Yamaguchi, Y., Ueno, T., Sano, K., & Boku, T. (2020). Performance Evaluation of Pipelined Communication Combined with Computation in OpenCL Programming on FPGA. In *Proc. int. symp. on parallel and distributed processing workshops (ipdpsw)* (p. 450-459). http://dx.doi.org/10.1109/IPDPSW50202.2020.00083

Giannozzi, P., Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., … others (2009). Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of physics: Condensed matter*, *21*(39), 395502.

Gorlani, P., Kenter, T., & Plessl, C. (2019, Dec). OpenCL implementation of Cannon's matrix multiplication algorithm on intel stratix 10 FPGAs. In *2019 International Conference on Field-Programmable Technology (ICFPT)* (p. 99-107). http://dx.doi.org/10.1109/ICFPT47387.2019.00020

Hao, X., Zhang, M., Sun, C., Tao, Z., Rong, H., Zhang, Y., … Liang, Y. (2023). Lasa: Abstraction and specialization for productive and performant linear algebra on FPGAs. In *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines (FCCM)* (p. 34-40). http://dx.doi.org/10.1109/FCCM57271.2023.00013

He, Z., Korolija, D., & Alonso, G. (2021, sep). EasyNet: 100 Gbps Network for HLS. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)* (p. 197-203). Los Alamitos, CA, USA: IEEE Computer Society. Retrieved from https://doi.ieeecomputersociety.org/10.1109/FPL53798.2021.00040 http://dx.doi.org/10.1109/FPL53798.2021.00040

He, Z., Parravicini, D., Petrica, L., O'Brien, K., Alonso, G., & Blott, M. (2021). ACCL: FPGA-Accelerated Collectives over 100 Gbps TCP-IP. In *2021 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC)* (p. 33-43). http://dx.doi.org/10.1109/H2RC54759.2021.00009

Intel DevCloud. (2023). [Online] https://www.intel.com/content/www/us/en/developer/tools/devcloud/overview.html. (Accessed: 2023-10-04)

IO500. (2023). [Online] https://io500.org/. (Accessed: 2023-10-09)

Jain, A. K., Ravishankar, C., Omidian, H., Kumar, S., Kulkarni, M., Tripathi, A., & Gaitonde, D. (2023). Modular and lean architecture with elasticity for sparse matrix vector multiplication on fpgas. In *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines (FCCM)* (p. 133-143). http://dx.doi.org/10.1109/FCCM57271.2023.00023

JupyterHub. (2023). [Online] https://jupyter.org/hub. (Accessed: 2023-10-04)

Kapil, V., Rossi, M., Marsalek, O., Petraglia, R., Litman, Y., Spura, T., … others (2019). i-pi 2.0: A universal force engine for advanced molecular simulations. *Computer Physics Communications*, *236*, 214–223.

Kenter, T., Shambhu, A., Faghih-Naini, S., & Aizinger, V. (2021). Algorithm-hardware co-design of a discontinuous Galerkin shallow-water model for a dataflow architecture on FPGA. In *Proc. Platform for Advanced Scientific Computing Conf. (PASC)* (p. 11). New York, NY, USA: Association for Computing Machinery (ACM). Retrieved from https://doi.org/10.1145/3468267.3470617 http://dx.doi.org/10.1145/3468267.3470617

Kikuchi, K., Fujita, N., Kobayashi, R., & Boku, T. (2023). Implementation and performance evaluation of collective communications using CIRCUS on multiple FPGAs. In *Proc. HPC Asia Workshops* (p. 15–23). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3581576.3581602 http://dx.doi.org/10.1145/3581576.3581602

Kühne, T. D., Iannuzzi, M., Del Ben, M., Rybkin, V. V., Seewald, P., Stein, F., … others (2020). Cp2k: An electronic structure and molecular dynamics software package-quickstep: Efficient and accurate electronic structure calculations. *The Journal of Chemical Physics*, *152*(19).

McCalpin, J. D., et al. (1995). Memory bandwidth and machine balance in current high performance computers. *IEEE computer society technical committee on computer architecture (TCCA) newsletter*, *2*(19-25).

McLay, R., Schulz, K. W., Barth, W. L., & Minyard, T. (2011). Best practices for the deployment and management of production hpc clusters. In *State of the Practice Reports.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2063348.2063360 http://dx.doi.org/10.1145/2063348.2063360

Menzel, J., Plessl, C., & Kenter, T. (2021, November). The strong scaling advantage of FPGAs in HPC for n-body simulations. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, *15*(1). http://dx.doi.org/http://dx.doi.org/10.1145/3491235

Meyer, M., Kenter, T., & Plessl, C. (2020). Evaluating fpga accelerator performance with a parameterized opencl adaptation of selected benchmarks of the hpcchallenge benchmark suite. In *2020 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC)* (p. 10-18). http://dx.doi.org/10.1109/H2RC51942.2020.00007

Meyer, M., Kenter, T., & Plessl, C. (2022). In-depth fpga accelerator performance evaluation with single node benchmarks from the hpc challenge benchmark suite for intel and xilinx fpgas using opencl. *Journal of Parallel and Distributed Computing*, *160*, 79-89. Retrieved from https://www.sciencedirect.com/science/article/pii/S0743731521002057 http://dx.doi.org/https://doi.org/10.1016/j.jpdc.2021.10.007

Meyer, M., Kenter, T., & Plessl, C. (2023, mar). Multi-FPGA Designs and Scaling of HPC Challenge Benchmarks via MPI and Circuit-Switched Inter-FPGA Networks. *ACM Trans. Reconfigurable Technol. Syst.*. Retrieved from https://doi.org/10.1145/3576200 http://dx.doi.org/10.1145/3576200

NHR Alliance. (2023). [Online] https://www.nhr-verein.de. (Accessed: 2023-10-04)

NVIDIA A100 Tensor Core GPU Architecture. (2023). [Online] https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf. (Accessed: 2023-10-04)

Opdenhövel, J.-O., Plessl, C., & Kenter, T. (2023). Mutation tree reconstruction of tumor cells on FPGAs using a bit-level matrix representation. In *Proc. Int. Symp. Highly-Efficient Accelerators and Reconfigurable Technologies (HEART)* (pp. 27–34). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3597031.3597050 http://dx.doi.org/10.1145/3597031.3597050

Pacemaker. (2023). [Online] https://clusterlabs.org/pacemaker/. (Accessed: 2023-10-04)

Petitet, A. (2004). Hpl- a portable implementation of the high-performance linpack benchmark for distributed-memory computers. *http://www.netlib.org/benchmark/hpl/*.

Prouveur, C., Haefele, M., Kenter, T., & Voss, N. (2023). FPGA acceleration for HPC supercapacitor simulations. In *Proc. Platform for Advanced Scientific Computing Conf. (PASC)*. New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3592979.3593419 http://dx.doi.org/10.1145/3592979.3593419

Ramaswami, A., Kenter, T., Kühne, T. D., & Plessl, C. (2021). Evaluating the design space for offloading 3D FFT calculations to an fpga for high-performance computing. *Applied Reconfigurable Computing. Architectures, Tools, and Applications. (ARC)*, *12700*, 285–294. http://dx.doi.org/10.1007/978-3-030-79025-7_21

Sano, K., Koshiba, A., Miyajima, T., & Ueno, T. (2023). ESSPER: Elastic and scalable FPGA-cluster system for high-performance reconfigurable computing with supercomputer fugaku. In *Proc. Int. Conf. on High Performance Computing in Asia-Pacific Region* (p. 140–150). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3578178.3579341 http://dx.doi.org/10.1145/3578178.3579341

Sato, M., Ishikawa, Y., Tomita, H., Kodama, Y., Odajima, T., Tsuji, M., … Shimizu, T. (2020). Co-design for A64FX manycore processor and "Fugaku". In *Proc. Int. Conf. on High Performance Computing, Networking, Storage and Analysis (SC)* (p. 1-15). http://dx.doi.org/10.1109/SC41405.2020.00051

Sheng, N., Tong, Z., Jiang, C., Ma, X., Yang, X., Li, H., … Zhang, Q. (2023). Microsecond simulation in a special-purpose molecular dynamics computer cluster. In *Proc. Int. Conf. on Bioinformatics and Computational Biology (ICBCB)* (p. 151-157). http://dx.doi.org/10.1109/ICBCB57893.2023.10246549

Slurm Node Features Plugin Programmer Guide. (2023). [Online] https://slurm.schedmd.com/archive/slurm-20.11.9/node_features_plugins.html. (Accessed: 2023-10-04)

Song, L., Chi, Y., Guo, L., & Cong, J. (2022). Serpens: A high bandwidth memory based accelerator for general-purpose sparse matrix-vector multiplication. In *Proc. Design Automation Conference (DAC)* (p. 211–216). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3489517.3530420 http://dx.doi.org/10.1145/3489517.3530420

Song, L., Guo, L., Basalama, S., Chi, Y., Lucas, R. F., & Cong, J. (2023). Callipepla: Stream centric instruction set and mixed precision for accelerating conjugate gradient solver. In *Proc. Int. Symp. on Field-Programmable Gate Arrays (FPGA)* (p. 247–258). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3543622.3573182 http://dx.doi.org/10.1145/3543622.3573182

Stewart, L. C., Pascoe, C., Sherman, B. W., Herbordt, M., & Sachdeva, V. (2021). Particle mesh ewald for molecular dynamics in OpenCL on an FPGA cluster. *arXiv:2009.12617*.

Treibig, J., Hager, G., & Wellein, G. (2010). Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In *2010 39th International Conference on Parallel Processing Workshops* (p. 207-216). http://dx.doi.org/10.1109/ICPPW.2010.38

Ueno, T., & Sano, K. (2023, mar). VCSN: Virtual circuit-switching network for flexible and simple-to-operate communication in HPC FPGA cluster. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, *16*(2). Retrieved from https://doi.org/10.1145/3579848 http://dx.doi.org/10.1145/3579848

Van Hirtum, L., De Causmaecker, P., Goemaere, J., Kenter, T., Riebler, H., Lass, M., & Plessl, C. (2023). A computation of D(9) using FPGA supercomputing. *arXiv:2304.03039*.

Wu, X., Kenter, T., Schade, R., Kühne, T. D., & Plessl, C. (2023). Computing and compressing electron repulsion integrals on FPGAs. In *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines (FCCM)* (p. 162-173). http://dx.doi.org/10.1109/FCCM57271.2023.00026

Xilinx Vitis Libraries. (2023). https://docs.xilinx.com/r/en-US/Vitis_Libraries.

XUP Vitis Network Example (VNx). (2023). [Online] https://github.com/Xilinx/xup_vitis_network_example. (Accessed: 2023-10-04)

Yoo, A. B., Jette, M. A., & Grondona, M. (2003). Slurm: Simple linux utility for resource management. In D. Feitelson, L. Rudolph, & U. Schwiegelshohn (Eds.), *Job Scheduling Strategies for Parallel Processing* (pp. 44–60). Berlin, Heidelberg: Springer Berlin Heidelberg.

Zeni, A., O'Brien, K., Blott, M., & Santambrogio, M. D. (2021). Optimized implementation of the HPCG benchmark on reconfigurable hardware. In L. Sousa, N. Roma, & P. Tomás (Eds.), *Euro-Par 2021: Parallel Processing* (pp. 616–630). Cham: Springer International Publishing. http://dx.doi.org/10.1007/978-3-030-85665-6_38