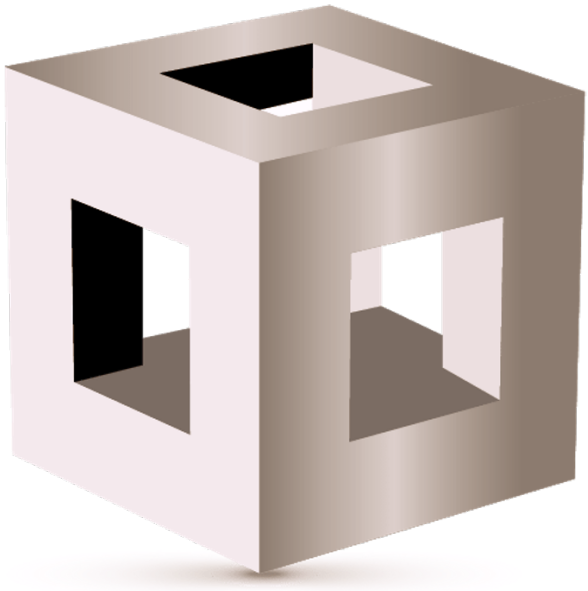
HORIZON
Topic: CL4-2022-DIGITAL-EMERGING-01
Digital and emerging technologies for competitiveness and fit for the green deal



OpenCUBE
101092984

D 2.1
Pilot System Architecture Design and Prototype Implementation Report

WP 2: Hardware Platform for the European Processor

Date of preparation (latest version): 2023-12-31

DOCUMENT INFORMATION

Deliverable Number	D 2.1
Deliverable Name	Pilot System Architecture Design and Prototype Implementation Report
Due Date	2023-12-31 (PM 12)
Deliverable lead	Semidynamics
Authors	V. Casillas (SiPearl), P. Marcuello (Semidynamics) J. Vaquero (Semidynamics), P. Deus (Semidynamics) H. Ponce (SiPearl), T. Mueller (HPE) P. Friese (TUM)
Responsible Author	Pedro Marcuello (Semidynamics) e-mail: `pedro.marcuello@semidynamics.com`
Keywords	OpenCUBE Pilot system, Rhea Processor RISC-V accelerator, Slingshot
WP/Task	WP 2/Task 2.1, 2.2, 2.3 and 2.4
Nature	R
Dissemination Level	PU
Final Version Date	2023-12-31
Reviewed by	Emanuele Danovaro (ECMWF) Stefano Markidis (KTH)

DOCUMENT HISTORY

Partner	Date	Comment	Version
KTH	2023-09-15	Skeleton version of the deliverable	0.1
KTH	2023-09-27	Updated version of the deliverable	0.2
All Partners	2023-12-05	Populated all sections and ready for internal review	0.3
SiPearl	2023-12-12	Update some missing parts	0.4
All Partners	2023-12-20	Rework conclusion and missing parts	0.5
KTH	2023-12-22	Update wording, terms	0.6
KTH	2023-12-31	Final cleanup, final version	1.0

Executive Summary

This deliverable describes the current state of deployment of the OpenCUBE pilot system based on the different components from the European Processor Initiative (EPI). Since the main components are not available yet, the system has been built with the appropriate replacements, as described in this deliverable, to facilitate the development of the related software tasks. In this way, the Rhea processor is to be replaced by Ampere Altra processors and the EPAC2 accelerator by the same FPGA scheme that is used as SDV in EPI-2 and EuPilot projects. All these components are connected through the high-performance Ethernet-based HPE Slingshot interconnect.

Contents

1 Introduction

The OpenCUBE project proposes to design a full-stack solution for a European Cloud computing blueprint that will be deployed on European hardware infrastructure. The different work packages of this project target the different components of the stack. In this way, Work Package 2 focus on developing the OpenCUBE pilot system based on the both hardware outcomes from the European Processor Initiative project, the Rhea processor and the EPAC2 accelerator, and it is the objective of this deliverable. On the other hand, the rest of the work packages focus on the operating system, the software stack (WP3), the middleware, the memory management and network software in WP4. Finally, WP5 establishes the requirements and the expected performance for the different kind of applications.

The objective of this deliverable is to describe the initial design of the pilot platform, including compute node architecture, network architecture, and system architecture and prototype implementation and benchmarking results. The first pilot system initially includes equivalents commercial components to the final ones, as it is stated in the proposal, in order to facilitate the early prototyping of the software stack since both hardware components are not available yet. Therefore, the first prototype described in this deliverable consists of ARM-based processor systems for Rhea, a FPGA-based hardware for the RISC-V accelerator and the high-performance Ethernet-based HPE Slingshot interconnect.

This deliverable is organized as follows: Section 2 describes the Compute Node architecture, including both the ARM and the EPAC2 emulation platforms, the Network Architecture and the Storage and the IO architecture. The Software components required for the execution of applications is described in Section 3. Section 4 details the current state of deployment of the OpenCUBE pilot system, including the security documentation agreed by the partners to access to the system. Finally, Section 5 summarizes the conclusions of the document and lists the next steps to be taken in the next year.

2 Hardware components

2.1 Computer Node Architecture

In this section we will give a brief introduction of the compute node architecture of OpenCUBE prototype. It will be based on ARM-based servers that have similar architecture of the SiPearl Rhea processor that will be deployed later in the project, and it includes the EPAC2 accelerator, the network topology and the rest of the components for the prototype cluster.

2.1.1 Rhea emulation

SiPearl Rhea processor is based on high performance, power efficient Arm Neoverse V1 cores, designed specifically for HPC workloads. Each Neoverse V1 core includes Arm Scalable Vector Extension (SVE) for high-performance double precision, single precision, BFloat16 and 8-bit integer performance, addressing the full range of HPC workloads including AI and Machine Learning. Incorporating in-package High Bandwidth Memory (HBM), Rhea delivers extraordinary compute performance and efficiency with an unmatched Bytes/Flops ratio. All the Arm Neoverse V1 cores, are connected through a Coherent on-chip network. This network includes PCIe and CXL to ensure smooth connection with accelerator and high speed low latency networks, which is mandatory for efficient HPC applications. This network also provides access to Last Level Cache (SLC, Arm terminology for Level 3 cache), DDR and HBM stacks to all the compute cores.

As Rhea platforms will not be available during the first phase of the project as a mitigation plan it was decided to deploy a first version of the prototype with an Arm based processor using same kind of architecture. HPE servers RL300 containing Ampere Altra processor were selected as they are based on Rhea previous generation of cores (Neoverse N1) and mesh (CMN600). Of course some features will not be testable on this prototype (like SVE and HBM) but all the porting effort of the software stack as well as applications can be performed without any issue. Those 1U servers with a single Ampere Altra processor can host several PCIe boards for network and accelerator. As a first step they were equiped with Slingshot network cards and then Xilinx FPGA boards will be used to emulate EPAC2 as described in the next section.

2.1.2 EPAC2 emulation

The OpenCUBE project accelerator hardware consists of a set of two Xilinx® Alveo U55C FPGA cards installed in ARM®-based server hosts. These FPGA cards were developed aiming at high-performance computational applications, making them uniquely suitable for the present project. Each card provides two 100 Gb/s capable QSFP28 connections for high-speed networking, a PCIe connection configurable as either a single 16-lane Gen3 link (1×128 Gb/s) or a dual link 8-lane Gen4 (2×128 Gb/s) configuration, and on-chip 16 GB of HBM2 (high-bandwidth memory).

By design, the Alveo U55C cards are equipped with a Xilinx XCU55C FPGA, containing an adequate number of logical resources (LUTs, Registers, BRAMs, URAMs) to deploy a bitstream consisting of one Semidynamics RISC-V *Atrevido*® Core with a 512-bit

Vector Unit on a Coherent Hub Interface (CHI) Network-on-Chip (NoC), with a Home Node (HN) and 1 MB of L2 Cache. Among several features, the RISC-V core provides support for high memory bandwidth via *Gazzillion Misses*®, RISC-V Vector ISA support for activation functions (SIMD instructions) and Zfh extension support for half-precision (smaller memory footprint, faster execution).

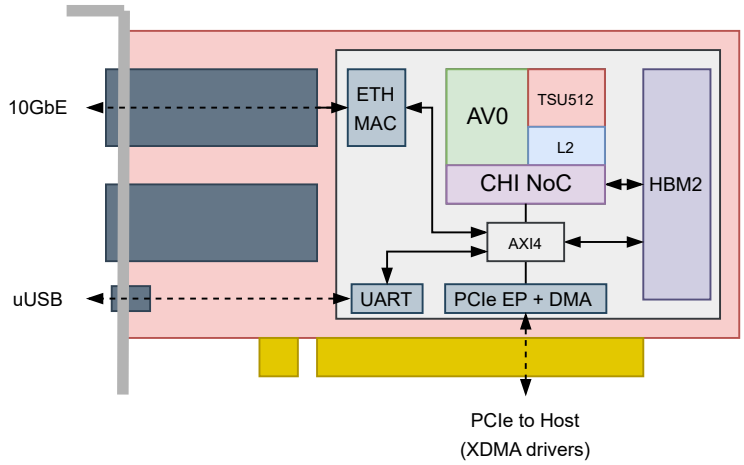


Figure 1: Simplified block diagram of the *Atrevido*® Core integration on the bitstream for the Xilinx Alvelo U55C FPGA card.

The bitstream architecture, depicted in Figure 1, was carefully designed to follow the project integration and interfacing requirements, as well as to maximize the connectivity paths between the *Atrevido*® Core, board peripherals and ultimately, the hosting server. The *Atrevido*® Core has direct access to the 16 GB of HBM2 through a dedicated AXI4 interface. In parallel, the same HBM2 memory is fully accessible from the server host side via PCIe. This low-level interfacing via PCIe allows for (i) loading data such as binary boot images into the board memory and (ii) driving/monitoring the core's specific interface signals relevant for operation kick-start. Alongside, the core is connected to a 10 GbE Network Interface, which enables (i) establishing terminal connections over *ssh* to the core and (ii) expanding storage through *NFS*. These features release the connectivity potential of the accelerator card as they extend access to any partner over the network. An additional terminal is implemented over *UART* and it is mostly intended for local access and maintenance operations. The same Micro-USB connector allows for establishing both the *ttyUSB UART* and JTAG programing connections.

2.2 Network architecture

There are three physically separated networks in the current pilot system:

- The hardware management network (HMN), a mostly $1\,\mathrm{Gbit\,s^{-1}}$ ethernet network connecting hardware-related components like the Base Management Controller (BMC), switch management interfaces and Power Distribution Unit (PDU).

- The client/customer access network (CAN), a $10\,\mathrm{Gbit\,s^{-1}}$ ethernet network used for shared home directory filesystem as well as OS management and SSH user traffic.
- The high-speed network (HSN), a $\geq 200\,\mathrm{Gbit\,s^{-1}}$ network based on HPE Slingshot for HPC traffic.

A network architecture for a full-scale deployment will depend on various factors, such as the size of a cluster or security considerations. Rather than trying to impose a customer network architecture we will instead try as best to accommodate for any possible scenario by only making the most basic assumptions.

This proposal therefore represents a minimal segmentation, accounting directly for the needs of the different physical parts. While all three network segments could be directly bridged (since they are based on Ethernet), their different speed characteristics as well as level of access to the underlying hardware, make it most likely that they will be routed in practice.

We are employing *virtual network segmentation* by grouping assigned IP addresses in a large enough network, and relying on DHCP where possible, which will make it easy to introduce additional subnets, should the need arise in the course of the project.

While the HMN and CAN are regular Ethernet-based networks, is the HSN based on HPE Slingshot, as an example of a High-performance Ethernet solution.

First, in standard Ethernet topologies, loops are not supported. The Spanning Tree Protocol (STP), respectively its extensions, the Rapid STP implement loop detection and automate disabling of redundant links. It does not support load balancing over links and is known to be slow to adapt when the topology changes (for example due to a switch or link failure). Load balancing is often implemented via the Link Aggregation Control Protocol (LACP), often called bonding or teaming, but is also supported by the Multi STP protocol.

This scales poorly in a High-Performance Computing (HPC) scenario as it leads to a tree-shaped architecture. To avoid bottlenecks as network utilization increases, links between switches must be scaled up accordingly. Furthermore, as the network grows, the number of hops between any two node increases as more layers must be added due to limited port capacity per switch. For latency there is no remedy.

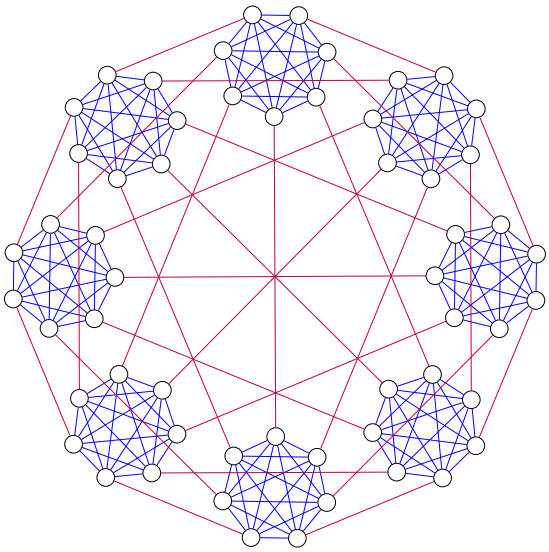
Furthermore, in a HPC network buffering is not an option since it will not be able to guarantee to maintain low latency, in particular not in busy networks as capacity gets saturated.

The problems mentioned above are crucial in an HPC network where guaranteed latencies are required to avoid stalling MPI collective functions which have to wait on the slowest connection to finish.

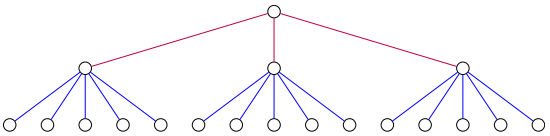
Slingshot solves all of the above issues with its Dragonfly topology, Adaptive Routing and Congestion Control.

2.2.1 Dragonfly Topology & Slingshot Network

A typical High-Performance Computing (HPC) application must distribute its state across many nodes and assumes the communication cost between arbitrary nodes to



(a) Dragonfly topology. Each node represents a switch. Purple edges are inter-group, blue are intra-group connections. Each group is connected to at least one other group. Edges can also be multiple aggregated links. Compute nodes can be connected to more than one switch in a group.



(b) Standard ethernet topology.

Figure 2: Network topology comparison.

be equal. HPC applications can easily saturate a connection but still require very low latency to avoid wasting cost intensive compute time. Acceleration of communication primitives on the level of MPI Collective Functions are needed to efficiently steer communication. Remote Direct Memory Access (RDMA) is again required to free up the CPU for other tasks and increase the performance.

HPE Slingshot's Dragonfly Topology as depicted in Figure 2a takes this into account by creating a hierarchical ring topology, effectively reducing the number of hops to a fixed constant [1]. By encoding the the switch and port numbers directly into MAC numbers, it furthermore forgoes the need for package routing tables and associated lookups. By enforcing a backstall at the NIC level in case of traffic congestion, it avoids costly buffer maintenance overhead on the switch side.

The management and configuration of such a network is then not automatically resolved on a peer-to-peer level, but rather centrally. In a Slingshot network, this is done by the Fabric Manager Node, an entity comprising of multiple services communicating with all the switches via an out-of-band (OOB) HMN.

2.3 Storage and IO architecture

2.3.1 Performance Implications

The storage architecture developed and deployed in OpenCUBE will utilise both NVMe-based local storage and network-based remote storage. Currently connected via the front-tend network, remote storage is going to utilise the Slingshot-based high-speed network.

Local storage will provide high-throughput and low-latency storage capabilities, which will however not match the storage size of the remote storage servers. Assuming an application with high I/O demands, performance will be additionally limited by the available local storage size. If the locally deployed storage is not sufficiently large, then the Slingshot-based remote storage must be used. Utilisation limits of the high-speed network however are shared between application-based communication, for example via MPI or RDMA, and storage-based communication. This may lead to a degradation of application-based communication during I/O intensive periods due to interference.

Mitigation of this performance degradation is possible either by sizing local storage in accordance to application requirements, or by adding a secondary storage-network. The monitoring architecture developed in Work Package 3 will provide insights into storage and IO utilisation.

3 Software components

3.1 Libraries, Drivers

OpenCUBE will provide basic libraries such as the linear algebra libraries, BLAS, Eigen and the library set Boost. Access to fabric-attached memory will be provided using the OpenFAM reference library. The high-performance computing communication interface MPI will utilise the Slingshot-based high-speed network using the network library libfabric. Access to hardware performance counters is given using the in-kernel `perf_events` subsystem, which also supports ARM platforms such as the Ampere Altra deployed in the OpenCUBE testbed.

3.2 Toolchain

3.2.1 ARM

Support for the ARM aarch64 target is readily available in both GCC 13 and LLVM/clang 18, therefore both toolchains will be made available to OpenCUBE users. Both compilers fully support OpenMP 4.5 and include partial support for OpenMP 5. GNU Fortran will be provided as a toolchain for Fortran-based applications. Specific versions of GCC and LLVM will be made available to OpenCUBE project with optimization done by SiPearl based on the different benchmarks results obtained during next periods.

3.2.2 EPAC2 & RISC-V

The accelerator part for the Opencube project is defined in the EPAC2 project, based on the RISC-V architecture. Due to delays in the EPAC2/Pilot projects, RISC-V accelerator will be accessible via a simulation infrastructure based on FPGA, as described in the section 2.1.2. As for initial steps, Semidynamics will provide the bitstreams to program the FPGA, the toolchain for cross-compiling from x86 into RISC-V binaries, and a set of initial libraries.

There are two tested and ready cross-compilers, both of them fully support RISC-V bit-manipulation, vector extensions, hardware counters, and half precision floats,

- Gcc 13.1 Toolchain. Including vector intrinsics support.
- Clang 18 Toolchain. Including vector intrinsics and vector half precision floating point support.

In order to support a development environment for such porting, implementation we are providing a Spike version that matches Semidynamics EPAC2 accelerator core. Qemu 8.0 might be used with full Linux support for the Vector 1.0 specification. We are including a set of highly optimized libraries for EPAC2 hardrware, with vector support and C standard string routines like strlib. For the AI scope, Semidynamics will supply specialized routines like convolutional layers, GEMM, activation functions, logistic activation, and maxpool. We will support as well quantization and other standards for data width reduction like brain float16 and brain float. Although bfloat is not part of the RISC-V

vector standard, *Atrevido*® and Semidynamics vector unit provides proprietary support for AI floating point formats such as brain floats.

4 Pilot Platform Demonstration

A Bill of Materials (BOM) of the relevant components can be found in Table 1, consisting of three groups of independent active hardware: network devices, compute devices and others (PDUs, Field-Programmable Gateway Arrays (FPGAs), etc.). We performed initial benchmarking on the OpenCUBE pilot system and present the results extensively in Deliverable D4.1: Initial system characterisation and middleware requirements, specification, and design, Section 2. As detailed in D4.1-Section 2, we used a set of benchmarks, including *iperf* for point-to-point Ethernet throughput, *netperf* for point-to-point Ethernet latency for Slingshot high speed network communication, and the OSU Microbenchmark Suite for both point-to-point and collective MPI communication, the MemoryLatency tool and Stream for memory subsystem, and the open-source core-to-core-latency benchmark for processor core topology.

Networking is performed via three switches, a specialised Slingshot switch, and dedicated standard Ethernet frontend and management switches. The Slingshot switch is required to build a Slingshot network, as listed in the proposal, which substitutes for a High-Performance Ethernet network. For the rest of the infrastructure, two switches were necessary to accommodate for the different connectivity methods: while the Out-of-band (OOB) management connections all feature a standard RJ45 connector for 1 GbitE, and can thus use cheaper cables, the 10 GbitE interfaces are usually equipped with SFP+ cages and require corresponding transceiver modules or the use of Direct Attach Copper (DAC) cables with integrated transceivers.

The compute side components consist of a Fabric Manager Node, specified according to official HPE Slingshot guidelines, and to be connected to the Fabric via a management port on the Slingshot switch. While its task is to configure and monitor the Slingshot network, it will also be used as the login node and host of shared home directories. The infrastructure nodes will be used as the Cray System Management (CSM) management servers for deployment, management and monitoring of the compute nodes. The compute nodes currently encompass 4 HPE RL300 servers with Ampere Altra ARM CPUs as a reference system to compare against the SiPearl Rhea Evaluation Board (EVB) to be delivered at a later date. They will also be used for direct development and testing on the ARM architecture. Note that the Ampere Altra is an ARM Neoverse N1 core and thus similar enough to the Neoverse V1 based Rhea CPU to allow this kind of substitute development.

Other devices beyond the networking and compute ones include a smart PDU that enables OOB power usage monitoring and full remote control. The system integrates a AMD/Xilinx Alveo U55C FPGA, which is discussed in section 4.2.

The installation process, from receiving the hardware, the hardware components, to the actual installation of the hardware into the dedicated rack, as well as the rack layout itself, can be seen in Figure 4.

For the initial prototype the network layout closely follows the available hardware components described at the beginning of this section.

We determined and evaluated three options for a network setup, as described in the following paragraphs. Note that, since the Slingshot network is always isolated over dedicated hardware, the following options mainly concern the logical separation of the

Table 1: Hardware Specifications

(a) Network devices

Name	Model	Ports
Management Switch	HPE FlexNet 5140 EI (R8J41A)	24x1 Gbit Ethernet RJ45 2x10 Gbit Ethernet SFP+ 2x10 Gbit Ethernet RJ45
Frontend Switch	HPE Aruba 3810M (JL075A)	12x10 Gbit Ethernet 2x Expansion Slots
Slingshot Switch	*Rosetta*	64x200 Gbit Slingshot

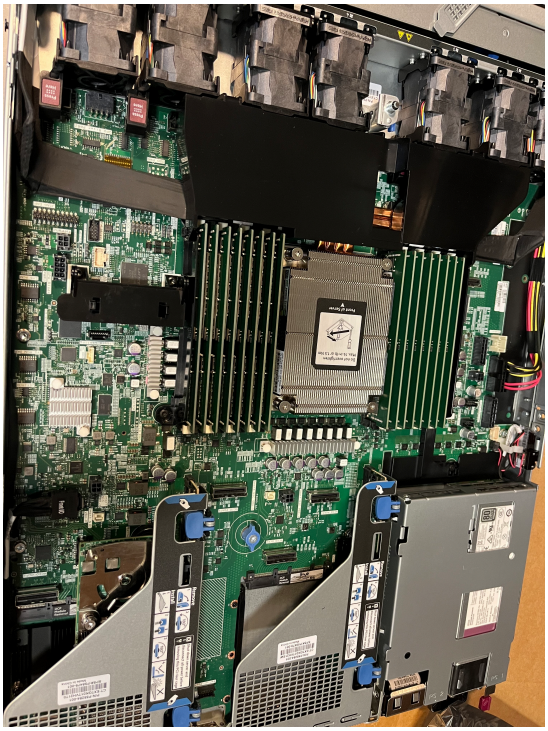
(b) Compute devices

Name	Qty.	Model	CPU	Memory	Storage	Network
FMN	1	HPE ProLiant DL325 Gen10 Plus v2	AMD 7443P 24C	256 GiB	4x3.84 TB NVMe	2x10 GbitE
Infra	2	HPE ProLiant DL325 Gen10 Plus v2	AMD 7313P 16C	512 GiB	1x480 GB SATA 1x960 GB SATA	2x10 GbitE 1x200 GbitS
RL300	4	HPE ProLiant RL300 Gen11	Ampere Altra	512 GiB	1x2 TB NVMe	2x10 GbitE 1x200 GbitS

(c) Other devices

Name	Description	Details
SmartPDU		
Xilinx/AMD Alveo FPGA U55C		Plugged into `infra-1`.

(a) Hardware packages.



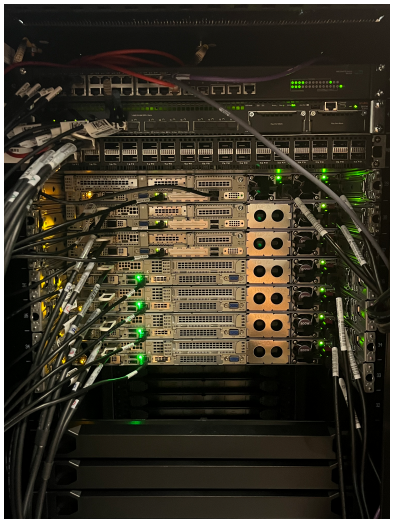
(b) RL300 w/ Slingshot adapter installed.

Figure 3: Phase I hardware ready for installation.

(a) Front

U	Description	U
42	Mgmt SW	42
41	Ethernet SW	41
40	Slingshot SW	40
39	FMN	39
38	Infra Server	38
37	Infra Server	37
36	RL300	36
35	RL300	35
34	RL300	34
33	RL300	33
32		32
31		31
30		30

(b) Unit occupation. Other units not shown are empty and available to OpenCUBE.



(c) Rear

Figure 4: Phase I rack installation.

three networks required: *uplink*, *frontend*, and *management*.

- We do not employ any Virtual Local Area Networks (VLANs) to separate the networks on the L2 level, but rely on physical interfaces instead.
- Each of the 10 GbitE ports of the Fabric Manager Node connect to one of the switches.
- An extra NIC (possibly with a transceiver) is added to the Fabric Manager Node to connect to the Data Center (DC) uplink.
- The Fabric Manager Node gets each of the three networks on one of three separate physical ports.

This setup is illustrated in figure 5. It also provides enough flexibility in regard to extensions or other deployment experiments (e.g. a separate storage network with Quality of Service (QoS) on L2 level).

For network and hostname naming as well as IP number assignment we are following CSM nomenclature (except for *xnames*):

- `HMN`: Hardware Management Network
- `CAN`: Client or Customer Access Network
- `HSN`: High-Speed Network
- `CN`: Compute Node

The detailed network assignment following that nomenclature is listed in table 2. To anticipate network segmentation to accommodate for test scenarios we are using large subnets out of the private allocations with holes. This permits easy mapping of server names to matching IP addresses and makes it possible to split machine groups into separate subnets if required without any relevant user-visible change.

4.1 Security documentation

Deploying a prototype within a collaborative cannot be done without doing a security assessment and reviewing access rules in order to protect each partner asset, that's why some work was done to define a security concept. Main ideas behind this security concept are first to protect some confidential assets that may come from some partners (SiPearl, Semidynamics or any other partner of the consortium) and used in this prototype and to avoid unexpected usage of this platform by nonmembers of the consortium. Idea is also to ease maintenance and access to this platform for the different partners, putting in place a complex setup will slow down usage of this platform from partners.

Security constraints will evolve during project lifecycle if partners provide confidential codes or assets to this prototyping platform. As of today, we do not consider this security concept is aimed to protect secrets or highly confidential data from any partners.

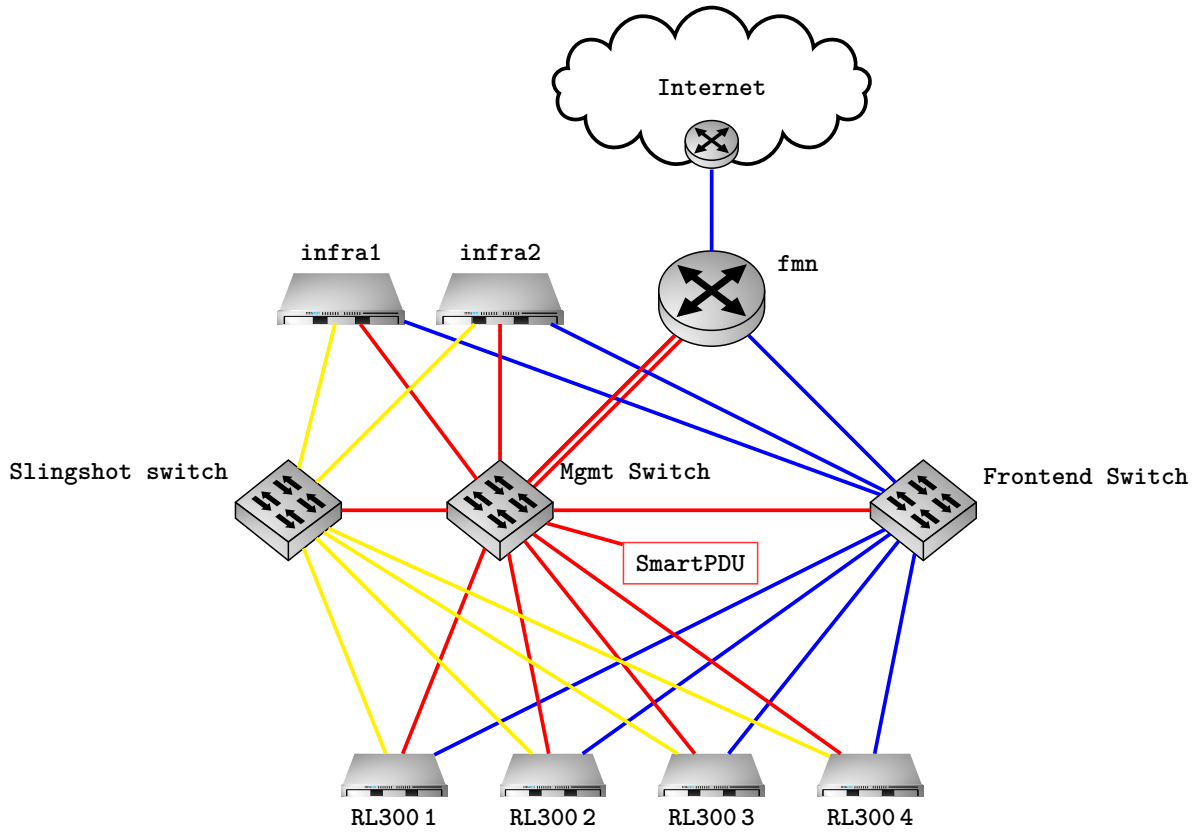


Figure 5: Prototype virtual network diagram. In red is the management network, blue the regular or frontend network and yellow the dedicated Slingshot network. The dual connection between the Fabric Manager Node and the Management Switch is the connection of the OOB management of the Fabric Manager Node itself and to give users on the Fabric Manager Node access to the management network itself.

Main security concept will be to define a login node as front-end to this cluster, that all partners will connect to via ssh. This login node will allow access to the rest of the cluster. Basic requirements defined below were agreed between the partners before deployment of this initial prototype:

- Access to login node will be done through ssh
- SSH configuration shall be compliant with ANSSI rules (NT_OpenSSH_en.pdf):
  - No root login
  - Public key only
  - Port shall be different than standard port (22)
- Access to login node will be filtered by IP.
- All cluster nodes shall be accessible from this login node.

Hostname	hmn.pt. 172.109.0.0/16	can.pt. 10.97.0.0/16	hsn.pt. 10.115.0.0/16
login.	172.109.0.1	10.97.0.1	
	172.109.0.2	10.97.0.2	
	172.109.1.1		
pdu.	172.109.1.2		
hmnsw1.	172.109.2.1		
cansw1.	172.109.2.2		
hsnsw1.	172.109.2.3		
infra1.	172.109.3.1	10.97.3.1	10.115.3.1
infra2.	172.109.3.2	10.97.3.2	10.115.3.2
cn1.	172.109.4.1	10.97.4.1	10.115.4.1
cn2.	172.109.4.2	10.97.4.2	10.115.4.2
cn3.	172.109.4.3	10.97.4.3	10.115.4.3
cn4.	172.109.4.4	10.97.4.4	10.115.4.4

Table 2: List of assigned IP addresses across networks. The intention of spreading the IP addresses is to make it possible to easily introduce additional subnets if needed and maintain an easy name to IPv4 address mapping.

- Firewall shall be configured on login node and only ssh shall be open.
- Access to external resource will be possible from login node and compute (GitHub, GitLab…) though a pxoxy to avoid direct access to internet (https and filtering will be setup).
- Login node configuration (SW stack) shall be reviewed before deployment by HPE, SiPearl and Semidynamics.

4.2 FPGAs

The system demonstration was performed using a single Alveo U55C card, installed at a PCIe slot available at the *infra1* host node. The Micro-USB connection, providing both the UART terminal and the JTAG links, was set to the *login1* host node, running Xilinx Vivado Lab Tools within a *Podman* software container over OpenSUSE Linux. In parallel, the card QSFP1 port was connected to a 10 GbE capable Network Switch to establish the Ethernet link between the accelerator card and the partner nodes.

At *login1*, the JTAG connection was verified and the U55C device was successfully detected by the Vivado Lab Tools. The U55C on-board memory was then flashed with a default bitstream containing a *i*) Xilinx PCIe Endpoint with DMA engine (same as the complete bitstream with *Atrevido*® Core) and *ii*) the HBM2 memory connected via AXI4. The purpose of this bitstream was not only to test the working status of this particular board unit but also to test the *i*) correct PCIe endpoint device enumeration, *ii*) Xilinx XDMA Linux device driver attachment and, consequently iii) the seamless exchange of

data between the *infra1* host and the card over PCIe. Once the on-board memory was flashed, the *infra1* was rebooted and the previously described steps were successfully achieved using a set of developed scripts for automatically testing the I/O access to the FPGA card. Furthermore, this approach based on a default bitstream (sharing the same PCIe endpoint configuration) presents the unique advantage of enabling to replace the bitstream running on the FPGA without rebooting the host and only implying a temporary device removal / rescan procedure from the Linux PCIe device tree. The steps required for the procedure were encoded into bash scripts and repeatedly used across the deployment tests.

Using the bitstream replacement technique, the FPGA was loaded with the configuration described in 2.1.2 and a RISC-V compatible Linux image was booted into the device. This process, automated using bash scripts and C++ XDMA I/O applications, makes use of the PCIe link to copy the Linux image into the FPGA board HBM2 and to configure the *Atrevido*® Core registers, triggering the Linux boot process. The Linux boot screen, available through the UART terminal at *login1*, allowed for verifying that the boot process was successful, to login to the device using a *root* account, and to verify the status of the Ethernet link. Finally, an *ssh* connection was established between *login1* host node and the U55C card, enabling to remotely access the card's Linux terminal and to exchange test files.

5 Conclusions and Next Steps

In this deliverable we have described the current state of deployment and the setting up process of the OpenCUBE pilot system. We have also described the architectural decisions taken in the design, especially related to the interconnection of the different components.

The first pilot system is now live and will be used by other work packages to port and benchmark their applications. Some developments for the OpenCUBE pilot are already planned in the next months. The first one will be to add some storage equipment to enable larger data sets for applications benchmarks. Then as Rhea platform will be only available by end of next period, we will integrate the first prototypes of Rhea platform (containing all the management part) that will be available in Q2 2024. This integration will allow fast deployment of systems once Rhea platforms will be available for the project.

On the RISC-V accelerator side, the current approach for executing applications in the FPGA is in standalone mode. Instead of that initial model for setting up and testing basic kernels and libraries, in the next reporting period we will focus in building the software tool-chain and providing the mechanisms to offload part of the applications from the general purpose cores to the accelerator. In this way, thanks to the feedback obtained from the rest of the partners in the project, we plan to work on the ONNXRuntime framework for executing HPC, AI and machine learning applications.

References

[1] John Kim et al. "Technology-Driven, Highly-Scalable Dragonfly Topology." In: *SIGARCH Comput. Archit. News* 36.3 (June 1, 2008), pp. 77–88. ISSN: 0163-5964. DOI: 10.1145/1394608.1382129. URL: https://doi.org/10.1145/1394608.1382129 (visited on 09/06/2023).

Glossary

Fabric Manager Node A set of services required to setup, manage and monitor a Slingshot network.. 10, 14, 17, 18

MPI Collective Functions Set of MPI functions involving 1:N, M:1 or M:N communication rather than only node to node. Ex. MPI Broadcast, Allgather, Scatter. Such functions must often wait on the slowest link to complete.. 10

Acronyms

BMC Base Management Controller. 8

BOM Bill of Materials. 14

CAN client/customer access network. 9

CSM Cray System Management. 14, 17

DAC Direct Attach Copper. 14

DC Data Center. 17

EVB Evaluation Board. 14

FPGA Field-Programmable Gateway Array. 14

HMN hardware management network. 8–10

HPC High-Performance Computing. 9, 10

HSN high-speed network. 9

OOB Out-of-band. 14, 18

OOB out-of-band. 10

PDU Power Distribution Unit. 8, 14

QoS Quality of Service. 17

RDMA Remote Direct Memory Access. 10

VLAN Virtual Local Area Network. 17