Pilot using Independent Local & Open Technologies

D18.1 Pilot System Specification

Version 1.0

Documentation Information

Contract Number	101034126
Project Website	www.eupilot.eu
Contractual Deadline	31.05.2022
Dissemination Level	Confidential
Nature	Report
Author	Albert Alarcón (Submer)
Contributors	Loïc Nasello, Jamal Al Chami (2CRSI), Iakovos Mavroidis (EXA)
Reviewer	David Vicente (BSC), Carlos Puchol (BSC)
Keywords	Facilities, Immersion Cooling, Specification

The European PILOT project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No. 101034126. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Italy, Switzerland, Germany, France, Greece, Sweden, Croatia and Turkey.

Change Log

Version	Description Change
V0.1	Initial specifications, pending integration
V0.2	Definitions and review of consistent use
V0.3	Improvements on introductions to sections, clarifications on dev systems vs. final systems, etc.
V0.4	More diagrams, cabled network solution
V0.5	Small edits by Carlos
V0.6	Small edits by Submer
V0.7	Intra-rack Network Solution 2CSRI & Submer
V1.0	Final document ready for submission

1. Executive Summary

The goal of this document is to describe the specifications, from chip modules to boards to chassis to datacentre deployment of The European PILOT (EUPILOT) system.

EUPILOT's system is an HPC OCP-based system, test rack, including host systems, accelerator systems, system power, intra-rack open network backplane and immersion cooling technology.

It also includes site survey specifications for deployment at the BSC (Barcelona Supercomputing Center).

2. Introduction

The EUPILOT project plans to deploy up to four cooling immersion *tanks*. A tank is equivalent to a conventional datacentre rack. Each of these tanks will contain a number of *host* and *EUPILOT accelerator systems*. Each host system will be connected to one or more accelerator systems.

Each of these accelerator systems contains up to 8 EUPILOT *accelerator modules*. Each of these modules is made up of one EUPILOT *chip* and its associated memory and some high-speed connectors for insertion in baseboard sockets.

The overall system is built from flexible components, enabling various ratios of servers to accelerators to be tested, as well as future research combining more accelerators.

EUPILOT accelerator systems are fairly host-agnostic and are integrated into a high-density OCP accelerator chassis that will be housed into a tank.

The figure below shows an EUPILOT system: a host/accelerator system pair and an overview of the EUPILOT Accelerator System components.

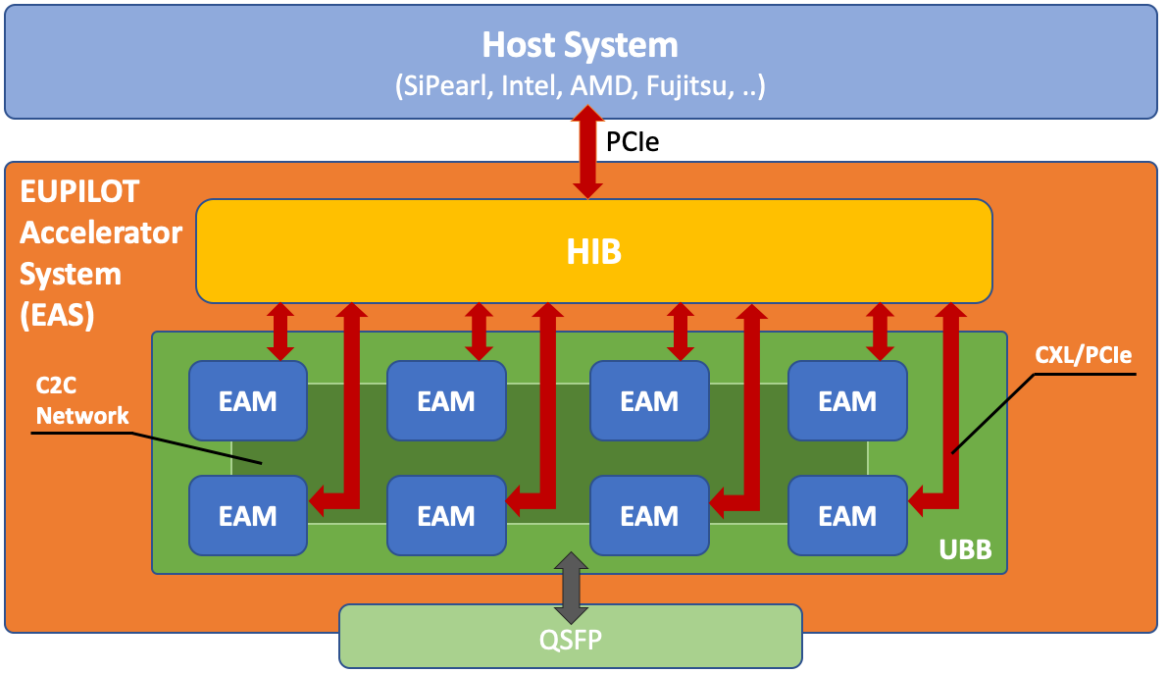

Figure 1: EUPILOT's Accelerator System Overview

3. European PILOT System Specifications

The aim of this section is to go deeper on the specifications of each component of the EUPILOT system. For the purposes of this document, we define an "EUPILOT System" to be a host server connected to one or more EUPILOT accelerator modules.

An EUPILOT system includes the following components:

- The EUPILOT chips, which are the accelerator chips of two variants. One variant is the multi-core MLS (Machine Learning Stencil) accelerator. The other variant is the VEC (Vector) accelerator. Below we will refer to "chip" regardless of which of the VEC or MLS varieties it may be. The chips are projected to have the same footprint.

- An EUPILOT accelerator module (EAM) is an OCP-compliant Accelerator Module (OAM) board and associated elements (heatsinks, stiffeners, connectors, …) which supports one EUPILOT chip and its associated memory chips. This is described in Section 3.1.

- The EUPILOT Accelerator System (EAS) (Section 3.2) is an OAI-compliant chassis mainly composed of a Universal Base Board (UBB) and Host Interface Board (HIB). The UBB is a carrier board that has up to 8 EAMs mounted on it. The HIB is used for communication between the EAMs and the host system.

- The Host System (Section 3.3) is a server that provides the base platform on which the host applications run and coordinate with the EAMs, through PCIe, in order to offload tasks to the EUPILOT accelerators.

The immersion cooling tanks (Section 3.4) contain several EUPILOT systems.

Host systems and EASs are expected to be connected one-to-one in pairs. The project expects each tank to typically contain 39OU (OpenU) rack units. Host servers will most likely have 1 OU and EUPILOT accelerator chassis will have 1 OU. Therefore, one tank may contain, at best, 19 EUPILOT systems, for a high degree of integration.

For development and testing, there may be other combinations of host/accelerator system solutions that may yield different densities in practice. There are also other considerations like networking hardware, power densities, etc. that may result in different densities.

3.1 EUPILOT Accelerator Module (EAM)

Prior to the design of the EUPILOT modules, partners will design a test board in order to test the IO connectivity of the EUPILOT *testchip*. This testchip will be much smaller and will be taped out well in advance of the accelerator chips proper. The purpose is to serve as a first validation vehicle of the new technology node used.

Following the testboard, partners will design the EUPILOT accelerator module (EAM), which will be an OAM-compliant device which will host an EUPILOT chip and associated memory chips, along with supporting components. The module will provide external connectivity, power and management to the EUPILOT chip. Connectivity interfaces are LPDDR (Low Power Double Data Rate) to memory, C2C (chip-to-chip) interfaces to other EUPILOT chips and CXL/PCIe to the external host. The PCB design of the accelerator module will follow the OAI-OAM standard v1.5[1]. Following this standard, the board will support two high-speed Molex connectors (Connector 0 and Connector 1) as shown in the figure below.

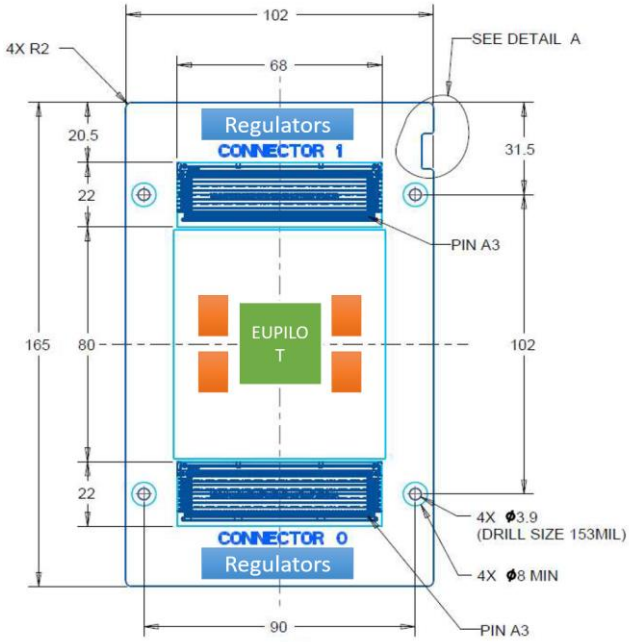

Figure 2: The EUPILOT OCP Acceleration Module (EAM)

The EUPILOT acceleration module includes one EUPILOT chip (green box), 4 LPDDR4 memories (orange boxes), two high-speed Molex connectors, regulators and peripheral circuitry.

The OAM connector 0 has the following interfaces:

1 https://www.opencompute.org/documents/ocp-accelerator-module-design-specification-v1p5-final-20220223-docx-1-pdf

- 54V/48V and 12V input power
- x16 SerDes to connect to host
- 3x16 SerDes for the C2C interface communication
- Other single-ended signals, like PRESNT#, SMBus, GPIOs etc.

The OAM connector 1 has the following interfaces:

- Power pins for 3.3V
- Other single-ended signals like JTAG, GPIOs etc.
- Up to 4 SerDes for the C2C communication or other purposes.

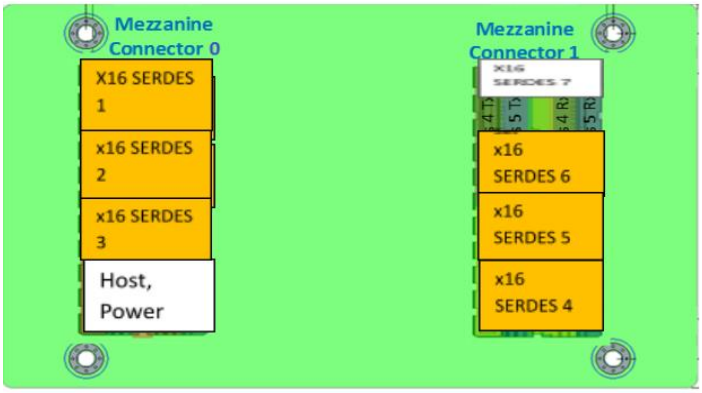

Figure 3: The two Molex Connectors of the accelerator module.

3.1.1 Port Mapping

The high-level planned floorplan of the EUPILOT (VEC) chip is shown in Figure 4.

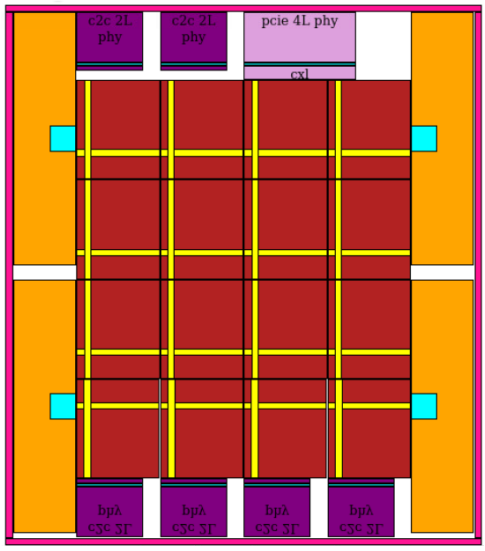

Figure 4: Floor plan of the EUPILOT chip.

The chip has seven ports, one 4-lane PCIe Gen 5 supporting the Compute Express Link (CXL) protocol (light pink block) and six dual-lane C2C links supporting 32 Gbps throughput per lane (dark pink blocks). The C2C link is a high-throughput low-latency link that transports CHI protocol. The processor dies include four LPDDR4 32-bit

memory controllers (large orange blocks on east/west sides), and sixteen RISC-V cores.

The 6 dual-lane C2C links and the quad-lane PCIe link will be routed to the two OAM connectors through the OAM PCB. The UBB and the HIB will provide the communication between the OAMs and the host as defined in the OCP specifications[2].

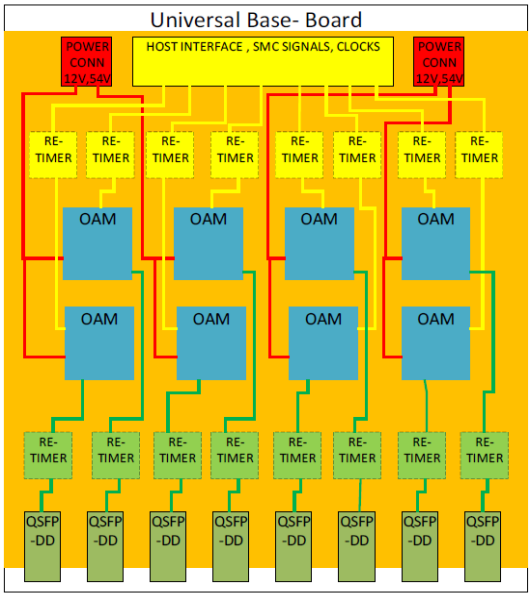

Figure 5: The UBB (Universal Base Board)

The yellow links and retimers shown in Figure 5 are used for the PCIe communication. The four PCIe lanes of the EUPILOT chip will be connected to the PETp/n pins of the OAM Connector 0. The yellow links provide communication between the OAMs (Connector 0 of the PCBs), the PCIe retimers on the UBB and the PCIe switches on the HIB.

The UBB also provides inter-OAM communication between the 8 modules (not shown in Figure 5). The UBB supports a fully-connected topology between the 8 modules as shown in Figure 6.

[2] https://www.opencompute.org/documents/universal-baseboard-design-specification-v1p5-final-20220223-docx-pdf

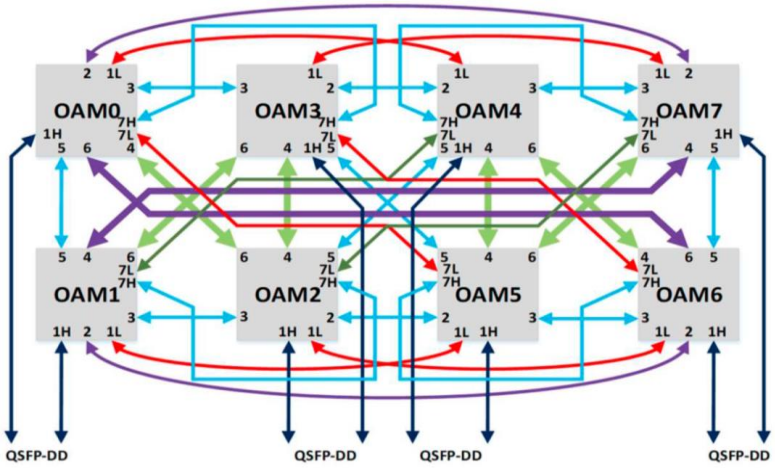

Figure 6: Fully-connected topology provided by the UBB.

The 7 links per OAM shown in Figure 6 correspond to the 7 SERDES of the module, shown in Figure 3, each one including 16 lanes. Links 1 and 7 are split into the low part (1L and 7L) and the high part (1H and 7H) in order to have four smaller 8-lane links (1L, 7L, 1H, 7H). Thus, in total we have 9 links per OAM (1L, 1H, 2, 3, 4, 5, 6, 7L, 7H) on the UBB. Since the EUPILOT chip has 6 C2C links and 2 lanes per link we cannot use all the links and all the lanes provided by the UBB.

One potential mapping of the six EUPILOT C2C links to the 9 links on the UBB is the following:

C2C link	UBB link	Lanes
1	1L	2 lower
2	1H	2 lower
3	3	2 lower
4	4	2 lower
5	5	2 lower
6	6	2 lower

Table 1. EUPILOT OAM port mapping

The resulting topology is shown in the figure below.

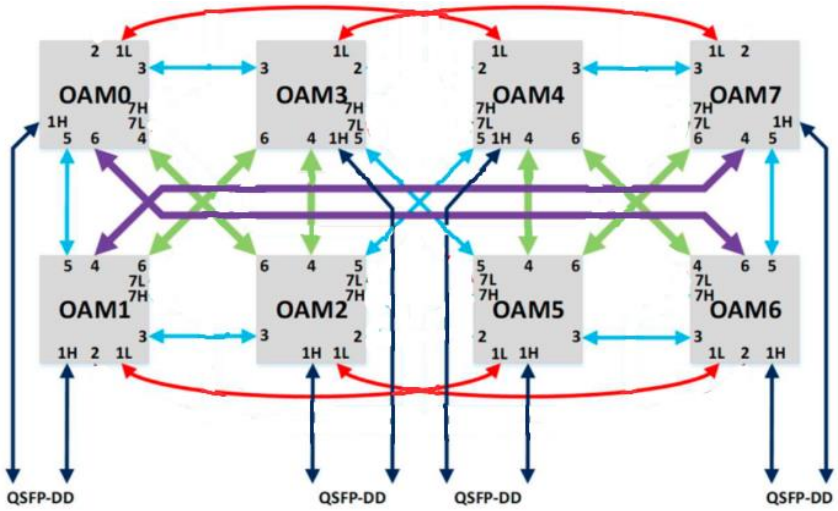

Figure 7: Two fully-connected quad OAMs.

3.2 OAM Chassis for EAS

3.2.1 Specifications for the testing solution

The expectation for the testing solution for a chassis/UBB/HIB/etc. to make up the EASs will be a commercial air-cooled solution available on the market, for example:

Model	Inspur MX1 21” OAI UBB System
Chassis	21” 3 OU rackmount
Dimensions	537W x 141H x 868.5D (mm)
Connection with Compute node	Up to PCIe Gen4 x32
OAM	Support Max 8pcs 48~54V OAM (up to 450W each)
	Support Max 8pcs 12V OAM (up to 350W each)
Power without OAM	1570W
PCIe Switch	Support PCIe Gen4 (100 lanes/chip)
PCIe Re-timer	Support PCIe Gen4 x16
Phy Re-timer	56Gbps PAM-4 or 10/28Gbps NRZ x16
Expansion Slots	Up to 4 x PCIe Gen4 x16 low profile standard card
BMC	AST2520
I/O	USB port for UART debug, RJ45 port for BMC dedicated, UID, PWR Button

3.2.2 Specifications for the final solution

EUPILOT partners (2CRSi) will develop a UBB, HIB, risers, power converters and other supporting components for the EAS system.

For risk mitigation, we can expect some commercial solutions to be available with similar specs to these:

Model	Next gen OAI
Chassis	21” 1 OU rackmount
Dimensions	537W x 141H x 868.5D (mm)
Connection with Compute node	Up to PCIe Gen5 x32
OAM	Support Max 8pcs 48~54V OAM (up to 450W each)
	Support Max 8pcs 12V OAM (up to 350W each)
Power without OAM	1570W
PCIe Switch	Support PCIe Gen5 (128 lanes/chip)
PCIe Re-timer	Support PCIe Gen5 x16
Phy Re-timer	56Gbps PAM-4 or 10/28Gbps NRZ x16
Expansion Slots	Up to 8 x PCIe Gen5 x16 low profile standard card
BMC	AST2520
I/O	USB port for UART debug, RJ45 port for BMC dedicated, UID, PWR Button

3.3 Host Systems

There are two planned solutions for host systems, one for initial testing and development and one for final deployment.

3.3.1 Specifications for the air-cooling development solution

This solution may be discrete, with cabling solutions appropriate for lab and “desktop” development. This is the expected specification of one such server.

Chassis	Model	OCtoPus 3, 1OU 3N
	Form Factor	21-inch 1 OU

	Dimension	873 x 177 x 43mm 34.5’’ x 6.9’’ x 1.69’’

Motherboard	CPU	2x Socket SP3 (LGA 4094) AMD EPYCTM 7xx2 and 7xx3 Series Processor family
	Chipset	System on chip
	Expansion slots	2x PCIe Gen.4 x24
	M.2 connector	2x M.2 NVMe (PCIe 4.0 x4) per node
	BMC	Aspeed® AST2500
Memory	Total Slots	48 (8-channel, 8-DIMM per CPU)
	Capacity per DIMM	Up to 64GB (with RDIMM modules) Up to 128GB (with LRDIMM modules) Up to 256GB (with 3DS RDIMM / LRDIMM modules)
	Memory Speed	Up to 3200 MHz
Storage	Internal type	Up to 4 x 2.5" PCIe x4 Gen4 U.2
Network	LAN	1x OCP NIC 3.0 (PCIe4.0 x16)
Front I/O	Ports	2x USB 3.2 Gen1Type-A 1x RJ45 ethernet 10/100/1000 Mbps management LAN 1x Mini Display or VGA
	Switch/LED	Per node: 1x Power button 1x UID button
Management Solution	Out of Band Remote Management	BMC Remote control based on Aspeed® remote management controller (Power Control Configuration, Chassis Identify, Boot Option, iKVM, BMC Account Configuration)
	TPM	1 x TPM header with SPI interface
Power Input	Type	OCP ORv3 48V

Figure 8: Testing Server

Figure 9: Testing Server

3.3.2 Specifications for the final solution

The expected solution for host systems for immersion cooling is with EUPEX hardware, as part of the collaboration to integrate EUPILOT and EUPEX systems. The host system specification details in this case will be forthcoming as the collaboration takes shape and the specifications of the EUPEX hosts become available.

There may be some air-cooled development servers needed for final integration prior to liquid immersion.

If the hardware from EUPEX does not materialize on time for reviews or demonstrations, The EUPILOT Project will demonstrate with conventional host servers similar or improved to the development solution specified above.

For example, if solutions with PCIe Gen5 or Gen6 interfaces are available for hosts systems, they will be preferable for deployment:

Chassis	Model	OCtoPus 3, 1OU 3N
	Form Factor	21-inch 1 OU
	Dimension	873 x 177 x 43mm 34.5'' x 6.9'' x 1.69''

Motherboard	CPU	2x Socket SP5 - AMD EPYC 5th Gen Turin family Or 2x Socket LGA7529 - Intel Xeon 6th Gen, Birch Stream family Or a Sipearl microprocessor-based system
	Chipset	System on chip
	Expansion slots	2x PCIe Gen.5 x16
	M.2 connector	2x M.2 NVMe (PCIe 5.0 x4) per node
	BMC	Aspeed® AST2600
Memory	Total Slots	48 DDR5 6800 or 6400 (12-channels, 12-RDIMM per CPU)
	Capacity per DIMM	Up to 96GB RDIMM modules supported Up to 192GB LRDIMM modules supported Up to 384GB 3DS RDIMM modules supported
	Memory Speed	Up to 6400 MHz
Storage	Internal type	Up to 4 x 2.5" PCIe x4 Gen5 U.3 or E.3S NVMe Drives or CXL Memory expansion
Network	LAN	1x Realtek RTL8211E for dedicated management GLAN (IPMI) OCP NIC 3.0 (PCIe x16 Gen 5)
Front I/O	Ports	2x USB 3.2 Gen1Type-A 1x RJ45 ethernet 10/100/1000 Mbps management LAN 1x Mini Display or VGA
	Switch/LED	Per node: 1x Power button 1x UID button
Management Solution	Out of Band Remote Management	BMC Remote control based on Aspeed® remote management controller (Power Control Configuration, Chassis Identify, Boot Option, iKVM, BMC Account Configuration)

	TPM	1 x TPM header with SPI interface
Power Input	Type	OCP ORv3 48V

3.4 Rack Unit

The immersion rack is designed to support the IT gear described on the section before being submerged and to contain the SmartCoolant.

The dimensions and capacity of the immersion tank are described below.

Specifications \ Model	**Submer Rack unit SPXL + (OCP 800mm CE version)**
IT heat dissipation capacity	100 kW or 50 kW fully redundant via 2x Cooling Distribution Units (CDU)
IT capacity	39OU (21 in. version)
IT hardware max depth	80 cm (not including 21 in. busbar and connector)
Dimensions (Lx W x H)	251 x 90 x 119 cm, with 195 cm max height when lid opens
Tank weight empty	767 kg / 1690.9 lb
Tank weight + SmartCoolant	1716.5 kg / 3784.2 lb
Footprint (L x W)	2.32 m²
Max floor load	1480 kg/m²
Operation air temperature range	5 °C to 40 °C
Secondary cooling technology	External Dry-Cooler / Adiabatic Dry-Cooler / Chiller
Max noise level	30 dBA
Lid type & Rack/tank colour	Opaque & Textured black (RAL 9004 alike)

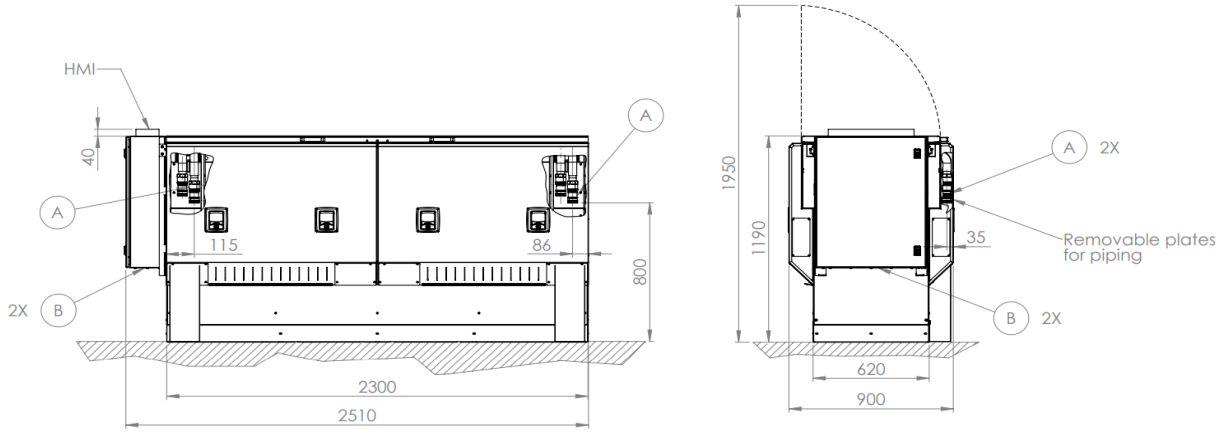

Figure 10: Illustration of the Submer Test Rack's basic layout
A: Water supply connection (inlet + outlet) (x2)
B: Power supply cable gland

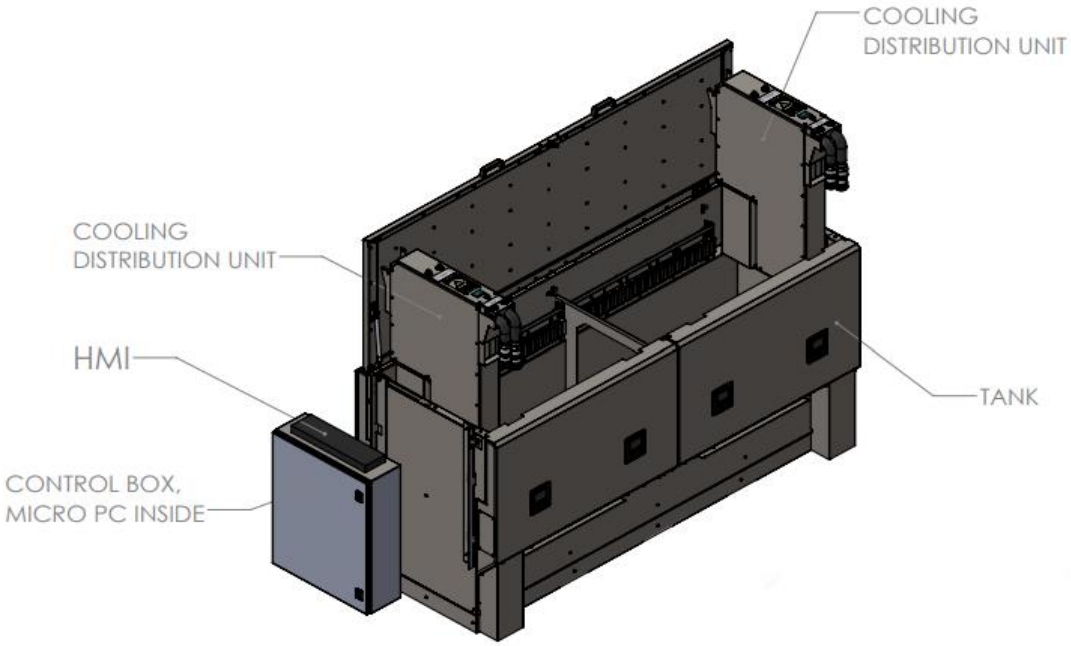

Figure 11: Illustration of the Submer Test Rack's 3D

Note: For the EUPILOT deployments the rack covers will be transparent glass.

The control box contains the power switchgear and the control hardware for the active components of the rack unit. It also has embedded the human machine interface (HMI).

3.4.1 System Power Rack unit

Power system spec/requirements of the Submer Rack unit

Max power consumption for 2x CDU	1.50 kW
Power supply type	3 Phase 400/230 Vac, 50 Hz
Power supply plug	2x Direct connection to grid 3P+E+N 16A (powering two CDUs) 2x Direct connection to grid 3P+E+N (TBD) (powering Power Shelves)
Max power consumption for Power Shelves	Up to 100 kW
Busbar type	Central busbar OCP V3
Power Shelf (powering IT equipment)	Up to 2x power shelves, each providing up to 50kW @48Vdc

Figure 12: Illustration of a 3P+E+N 16A IEC 60309 industrial connector

3.4.2 Rack Unit Cooling specification

The Cooling Distribution Unit (CDU) is the component that allows the flow of the SmartCoolant through the internal circuit of the Rack unit (see figure 14). It contains the circuit components necessary to keep the SmartCoolant set point temperature by passing the heat from the coolant to the water. Its position in the tank depends on the model.

Water requirements of the Test Rack	
Quality specification	As per ASHRAE Facility Water System (FWS) standard
Flow	18 m^3/h per tank (9m^3/h) x 4 → 72m^3h
Head loss in CDU	1.3 bar @rated flow
Inlet temperature	18 °C to 32 °C Note: BSC facilities will need to adapt to those requirements. Currently they have as inlet temperatures two different temperatures. 17º or 35º

Outlet temperature	Expected 5 °C increase with 100 KW capacity load
Supply connections	4x Zero-leak quick water coupling G 1 1/4" BSPP female

Figure 13: Inlet and outlet water connection at the immersion tank unit

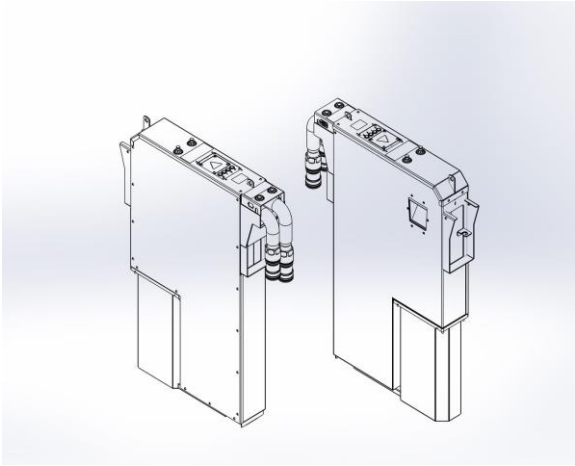

Figure 14: Rack unit cooling distribution unit (CDU).

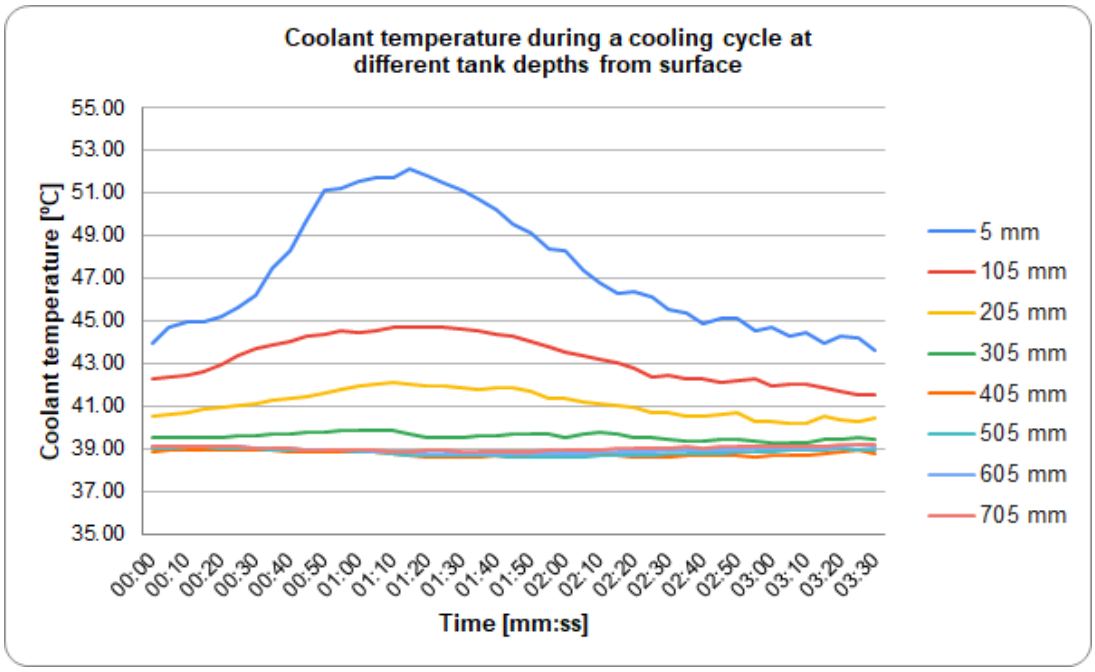

Figure 15: Coolant temperature during a cooling cycle

3.4.3. Intra-rack Network Solution

EUPILOT partners (2CRSI and Submer) are expecting 3 possible solutions for the intra-rack host networking to provide 200Gb speed per port, based on open project design (OCP or Open19).

Herewith the planned technical specifications:

Ports	1 port (12 Differential pairs) per Network Zone per OU (48mm). If both zones are populated, 2 ports per OU
	Cable-based or PCB backplane (for ex. Samtec ExaMAX)
Speed	Initial design speeds up to 200Gb per port
	PAM4 (112Gbps) support per differential pair
Switching	QSPF56 harness, Patch panel with QSPF56 ports or blind-mate central header aggregation for network switching / management hardwarepair

2CRSI in collaboration with Submer are working in 3 possible approaches for the Intra-Rack networking:

- ***Submer tank's Backplane***

Submer already deployed a design that includes a backplane with 2 network ports for each OU, this is almost discarded as there are several challenges for the chassis with

3 nodes as there is no possible to add a 3rd connector in the centre for the power busbar and also the tolerance with the brackets for inserting the nodes might cause issues to centre the connections in the nodes and the backplane.

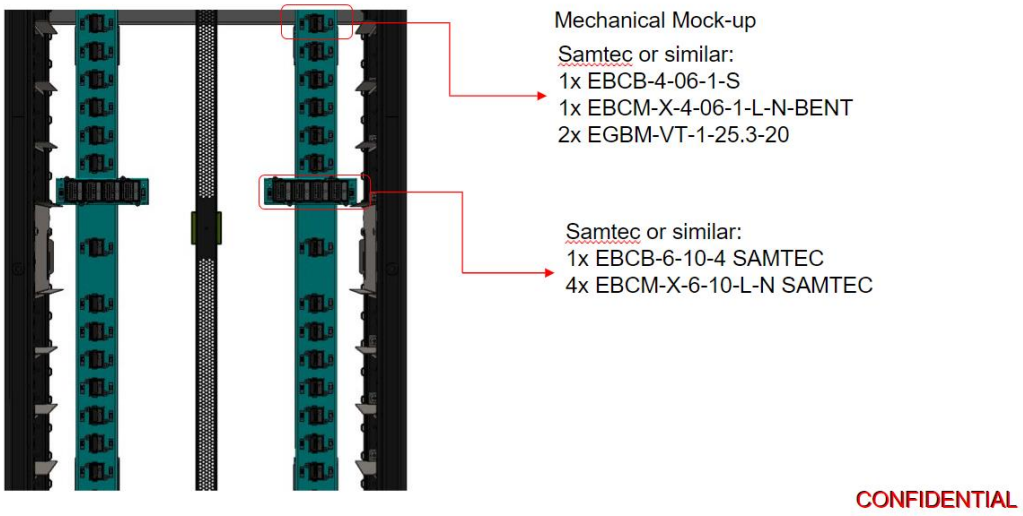

Figure 16: Backplane mechanical mock up.

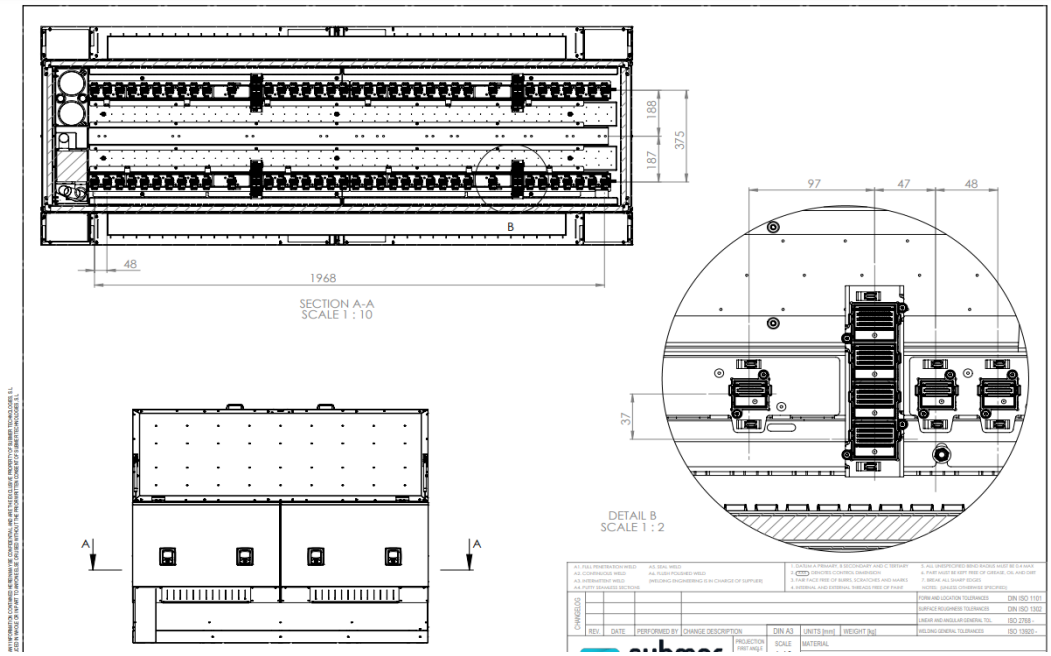

Figure 17: Backplane layout at Submer Rack unit XL

- ***OPEN19 Harness***

Using existing Open19 design connected in the chassis of the nodes and a harness from the bottom to the top of the tank and with QSFP56 connectors to the switches.

Figure 18: Open19 design layout

- ***Customized PAM4 connection in the bottom of the chassis of nodes***

2CRSI in collaboration with Submer will deploy a custom cable harness with a panel mounted ExaMAX2® or similar connector on chassis of the node side, to QSFP-DD connectors on switch side.

Figure 19: ExaMAX2® backplane and chassis connector

3.5 Software Specifications

Examon is a highly scalable framework for the performance and energy monitoring of HPC servers. We have chosen Examon to monitor the deployment of the EUPILOT systems.

3.5.1 Examon data model framework

This section describes the current implementation of Examon, its data model and data access interface.

○ *Architecture*

The architecture of Examon is composed of different layers, each of them with multiple components. The integration of different data sources is handled by the compositional nature of the infrastructure, where new components can be added seamlessly provided that they respect the correct data formats. The cornerstone of Examon is the middleware layer provided by MQTT brokers, which are the receivers of the data generated by the low-level plugins. On top of the MQTT brokers there is the data storage layer, where the data is uniformly formatted. From the storage layer the data can be fed to the high-level applications layer, by exploiting a client that exposes the underlying data collected and stored within Examon.

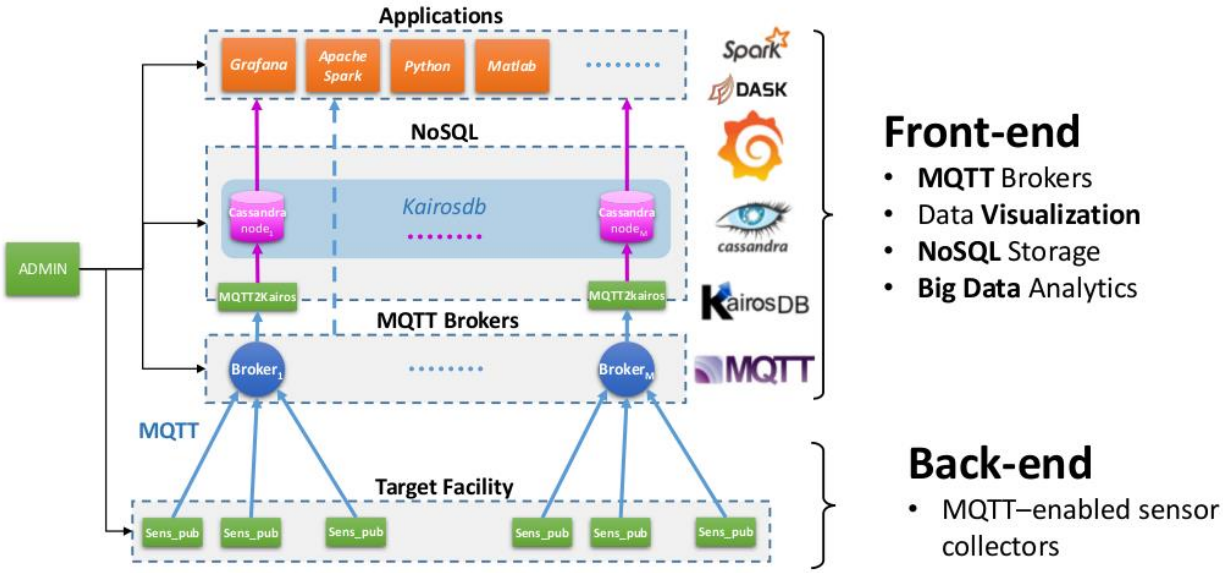

Figure 20: Software architecture

○ *Sensor Collectors*

These are the low-level components having the task of reading the data from the several sensors scattered across the system and delivering them, in a standardised format, to the upper layer of the stack. These software components are composed of two main objects, the MQTT API and the Sensor API object. The former implements the MQTT protocol functions and it is the same among all the collectors while the latter implements the custom sensor functions related to the data sampling and is unique for each kind of collector. Considering the specific sensor API object, we can distinguish collectors that have direct access to hardware resources like IPMI, PMU units in a CPU, and collectors that sample data from other applications (i.e Ganglia and Nagios) and batch schedulers (i.e. Slurm).

○ *Communication layer*

The framework is built around the MQTT protocol. MQTT implements the "publish-subscribe" messaging pattern and requires three different agents to work: (i) The "publisher", having the role of sending data on a specific "topic". (ii) The "subscriber",

that needs certain data so it subscribes to the appropriate topic. (iii) The "broker", that has the functions of (a) receiving data from publishers, (b) making topics available to subscribers, (c) delivering data to subscribers. The basic MQTT communication mechanism is as follows. When a publisher agent sends some data having a certain topic as a protocol parameter, the topic is created and available at the broker. Any subscriber to that topic will receive the associated data as soon as it is available to the broker. In this scenario, collector agents have the role of "publishers".

- ***Storage layer***

Examon provides a mechanism to store metrics mainly for visualisation and analysis of historical data. We use a distributed and scalable time series database (KairosDB) that is built on top of a NoSQL database (Apache Cassandra) as a back-end. Examon implements a specific MQTT subscriber (MQTT2Kairos) to provide a bridge between the MQTT protocol and the KairosDB data insertion mechanism. The bridge leverages the particular MQTT topics structure of the monitoring framework to automatically form the KairosDB insertion statement. This gives multiple advantages: first, it lowers the computational overhead of the bridge since it is reduced to a string parsing operation per message; secondly, makes it easy to form the database query starting only from the knowledge of the matching MQTT topic; lastly, it decouples the transport layer from the storage layer making it easy to migrate to new data storage systems.

- ***Applications Layer***

The data gathered by the monitoring framework can serve multiple purposes, as presented in the application layer. Data can be visualised using web-based tools or, for example, machine learning techniques can be applied to build predictive models or online fault detection algorithms. The Examon framework provides a convenient software component (examon-client) which implements a consistent and unique interface between the heterogeneous storage layer and the applications.

- ***Data model***

Examon adopts a non-relational, hierarchical data model, which is essential for dealing with a huge amount and variety of data.

At the heart of the data model is the concept of metric. The metric is essentially the physical entity (such as a sensor) or abstract entity (such as a log line generated by a running application) that generates a value (data). The value can be a number or a string or a whole line of text and is associated with a timestamp, i.e. the instant in time when the value was collected. Finally, each metric can have one or more tags, in the form of key-value pairs, that describe in detail the properties of the metric.

In Examon, the tags are used to define extra information about the sensor and some special mandatory properties of the protocol.

The data model resembles a hierarchical tree. In the Figure are shown the relevant sections.

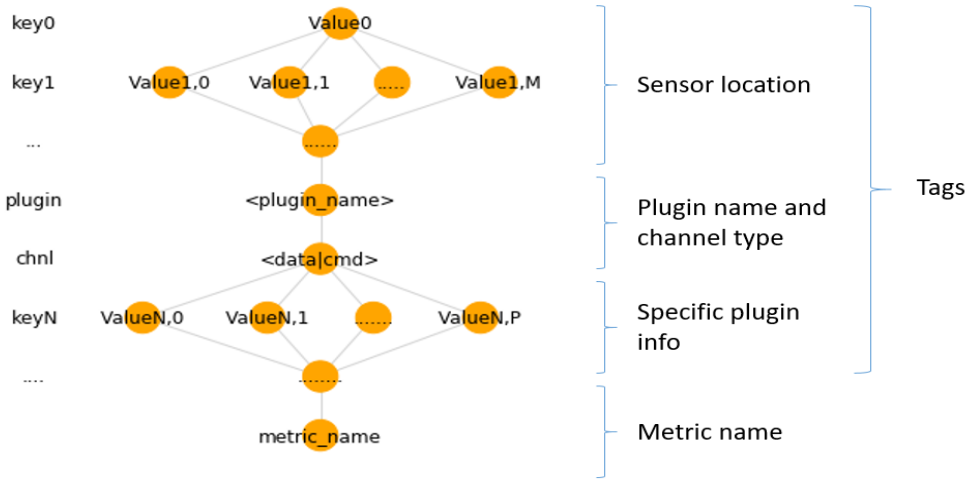

Figure 21: Data Model

- Sensor location: (Mandatory) it is a free hierarchical sequence of key/value couples used to locate the data source. At least one couple should be present.
- Plugin name: (Mandatory) it is the name of the data collector agent (plugin) which acquires data from this sensor
- Channel type: (Mandatory) specifies the type of channel
- data: metrics values sent by the plugin
- cmd: commands sent to the sensor/plugin
- Specific plugin tags: (Optional) additional free key/values couples to add custom plugin attributes that are unique for this data source.

The data source or a sensor is effectively defined when it is possible to uniquely determine a complete path within the tag hierarchical tree, from top to bottom. This means that in Examon, a sensor can only have one value for each of its tags.

Currently, to enter data within Examon you must connect to the transport layer (MQTT Broker) and send messages in a specific format.

- ***Transport***

This data model fits well with the MQTT protocol of the transport layer. Indeed, the tags and metric name define the topic, while the value and timestamp are part of the payload.

Topic	Payload
<key0>/<value0>/<key1>/<value1>/ ... /keyN/<valueN>/<metric_name>	<value0>;<timestamp0>
<key0>/<value0>/<key1>/<value1>/ ... /keyN/<valueN>/<metric_name>	<value1>;<timestamp1>
...	...
<key0>/<value0>/<key1>/<value1>/ ... /keyN/<valueN>/<metric_name>	<valueM>;<timestampM>

Figure 22: MQTT protocol of the transport layer

- ***Storage Layer***

In the storage layer the Examon data model is mapped to the Cassandra table schema. A table in Cassandra is a collection of rows. Each row is identified by a primary key. KairosDB defines primary keys in Cassandra primarily as a concatenation of the metric_name and tags and are used to index and search data. The values and timestamps on the other hand represent the columns of the row.

<metric_name>+<timestamp_base>+<data_type>+<key0>+<value0>+ ... + <keyN><valueN>	timestamp_off0	timestamp_off1	...	timestamp_offM
	<value0>	<value1>		<valueM>

Figure 23: Cassandra table schema

- ***Query language***

The data stored by Examon can be retrieved by direct queries to the Cassandra and KairosDB modules. However, data in Cassandra is stored in de-normalized form. While this simplifies the scalability of the system and increases its performance, it also adds complexity to data management from an application perspective.

In order to simplify the development of applications based on Examon data, a uniform entry point for all the data was developed and named examon-client. It will be described in the following section.

Examon-client This client makes it possible to query metrics from the Examon database. It is based on a pluggable interface and, in the current implementation, the KairosDB REST API is implemented. Examon-client is a python package that enables uniform access and analytics on Examon data; it offers a SQL-like query language for ease of use. Examon data can be accessed locally using the Pandas interface or in a distributed fashion (for heavy workloads) using Apache Spark or Dask.

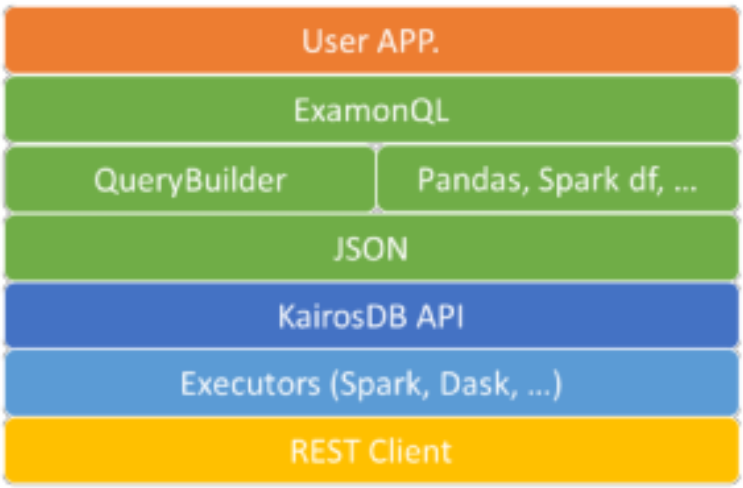

Figure 24: Query language

The installation instructions for Examon-client can be found in the online repository; it can be installed locally (no root permissions needed). After having installed the module, the first thing to do is importing the client:

```
from examon.examon import Examon
```

Then an examon instance must be created:

```
KAIROSDB_SERVER = '127.0.0.1'
KAIROSDB_PORT = '8083'
USER = ''
PWD = ''
ex = Examon(KAIROSDB_SERVER, port=KAIROSDB_PORT, user=USER, password=PWD,
verbose=False)
```

Username ("USER") and password ("PWD") must be requested by contacting Examon maintainers.
After the examon instance has been created, a query can be made. For instance, if we want to obtain a time-series containing the values of the temperature of node "node067" between "10-10-2017 17:09:00" and ""10-10-2017 21:10:00", grouped by core and with no aggregation (e.g. without averaging the values), the following query can be used:

```
tstart = "10-10-2017 17:09:00"
tstop = "10-10-2017 21:10:00"
metrics = ['temp']
tags = {'node': ['node067']}
groupby = {'name':'tag', 'tags':['core']}
aggrby = None
data = ex.query(tstart, tstop, metrics, tags, groupby=groupby,
aggrby=aggrby)
```

The json data returned by the query is converted to a Pandas Dataframe. Currently, the client returns queried data in two basic layouts: as table (narrow table) and as time series (wide table).
The default format (.df_table) is a generic narrow table. One row of this table represents a single data point. The columns are the corresponding tag values and the headers are the tag names. They match exactly the corresponding Examon MQTT topics key/value pairs.

```
# show results preview
data.df_table.head()
```

	chnl	cluster	core	name	node	org	plugin	timestamp	value
0	data	galileo	0	temp	node067	cineca	pmu_pub	2017-10-10 17:09:00	44
1	data	galileo	0	temp	node067	cineca	pmu_pub	2017-10-10 17:09:02	43
2	data	galileo	0	temp	node067	cineca	pmu_pub	2017-10-10 17:09:04	45
3	data	galileo	0	temp	node067	cineca	pmu_pub	2017-10-10 17:09:06	45
4	data	galileo	0	temp	node067	cineca	pmu_pub	2017-10-10 17:09:08	45

The generic tabular format returned by the query can be converted in a time series format dataframe (wide table). The first column (index) is the time vector while the remaining columns are the data vectors. A multi-index header is automatically generated from the query metadata.

```
data.to_series()
data.df_ts.head()
```

name	temp															
core	0	1	10	11	12	13	14	15	2	3	4	5	6	7	8	9
timestamp																
2017-10-10 17:09:00	44	46	49	44	46	38	43	39	48	46	47	43	44	46	46	45
2017-10-10 17:09:02	43	46	49	46	46	39	43	38	49	46	47	43	45	45	48	47
2017-10-10 17:09:04	45	46	49	45	46	39	42	38	49	48	47	43	46	45	47	46
2017-10-10 17:09:06	45	46	49	46	47	39	42	38	49	44	47	43	45	45	46	46
2017-10-10 17:09:08	45	45	48	46	46	39	42	38	49	47	47	43	46	46	47	46

The resulting time series can be visualized using the plot method provided by Pandas (matplotlib)

```
data.df_ts \
    .interpolate(method='time') \
    .plot(marker='.') \
    .legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

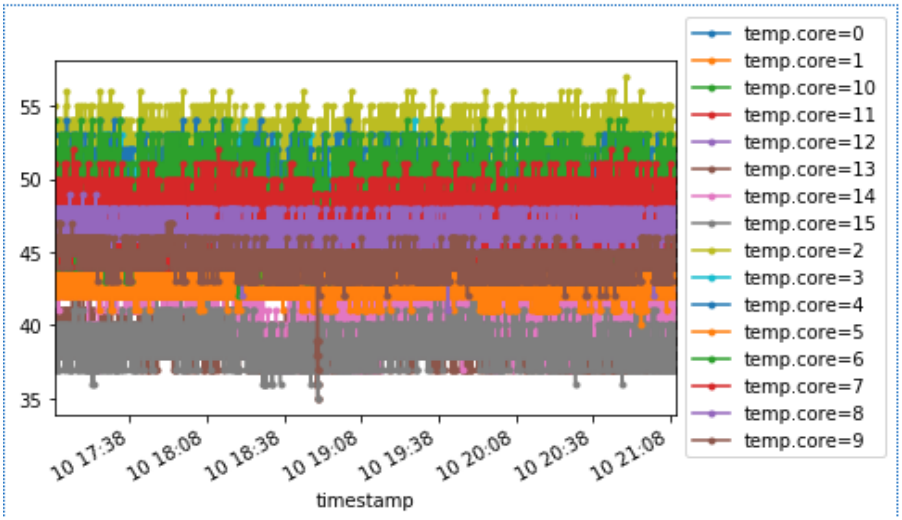

Figure 25: example of visualisation of plot method

Both tabular (.df_table) and time series (.df_ts) layouts can be exported to a file (.csv). The output format can be selected using the "shape" parameter.
The same query shown earlier can be also built and executed with an SQL-like language. In this case we have to rely on the Examon Query Language module, ExamonQL. First, an ExamonQL object has to be created:

```
sq = ExamonQL(ex)
```

Now we can define the SQL-like query:

```
data = sq.SELECT('*') \
    .FROM('temp') \
    .WHERE(node='node067') \
       .TSTART(tstart) \
       .TSTOP(tstop) \
       .groupby('core') \
       .aggrby('none') \
       .execute()
```

The returned object is a Pandas dataframe.

- ***Distributed query execution***

To perform heavy workloads, a distributed approach is more suitable. In this case, Examon-client offers a Spark/Dask interface allowing efficient data analysis on data streams.In this example we will use the Apache Spark execution engine. From the code point-of-view, we need again to obtain an ExamonQL object:

```
sq = ExamonQL(ex)
```

The query can be defined in the usual way and syntax, with the only difference being that now a query plan can be obtained by using the 'get_query()' method. The query plan is a collection of jobs that are going to be executed by the Spark workers.

```
query = sq.SELECT('*') \
    .FROM('temp') \
    .WHERE(node='node067') \
       .TSTART(tstart) \
       .TSTOP(tstop) \
       .groupby('core') \
       .aggrby('none') \
       .get_query()
```

The query plan has to be passed to the Spark executor:

```
from examon.executors import ExSpark

sp = ExSpark(ex)
```

Finally, the RDD is defined by processing the query plan, in this case, using 8 Spark workers :

```
rdd = sp \
     .spark(sc) \
     .query(*query, batch_size=30*60*1000) \
     .to_rdd(npartitions=8)
```

More examples can be found on the online repository.

- ***Data Visualization***

Another way to access the data is by exploiting the visualization tool integrated in Examon as a high-level component, precisely Grafana[3]. With Grafana live visualization of different data sources can be obtained, ranging from information about completed jobs, system services status and HW-sensors' measurements. A wide set of views and dashboard have been already prepared and can be accessed by using the IP address and port of the server hosting the service -- attainable by contacting Examon-client maintainers.

[3] https://grafana.com/

For instance, selecting a node and a time interval (e.g., the last 5 minutes), the different metrics can be visualised as time-series and/or aggregated values (average, max/min, variance, etc).

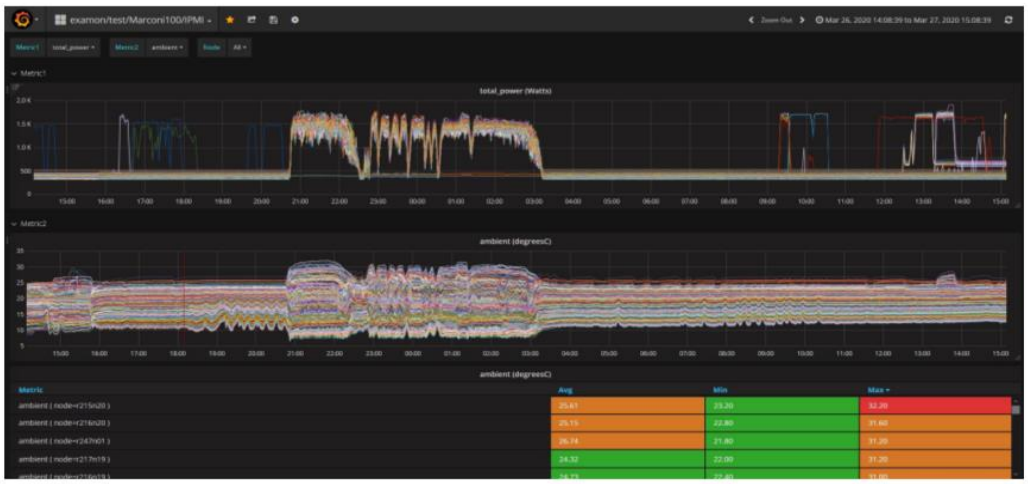

Figure 26: Example of data visualization

Alternatively, Grafana can be used to gain an overall view of all the racks of the data centers, with many potential benefits, among them identifying nodes with anomalous thermal load or workload imbalance.

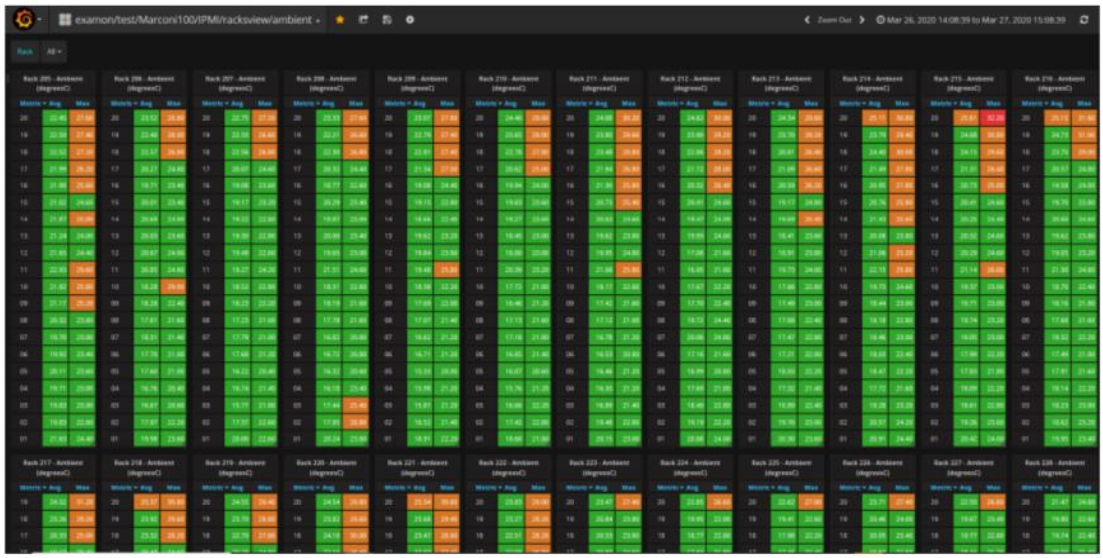

Figure 27: view of all racks in Grafana

- ***System Configuration***

Examon has to be installed in the cluster management nodes. This means that both the frontend (message bus, database and visualization) and the backend (collectors) are installed entirely on the cluster management system. In particular, the frontend section is typically installed on the login node while the collectors are installed on all the compute nodes. The metrics monitored are implemented as plugins.

- ***In-band plugins:***

This family of plugins executes as much as necessary within the compute nodes:

2. pmu_pub: this plugin is dedicated to collecting data about ARMIDA CPUs and includes their temperatures, powers, voltages and frequencies
3. nvidia_pub: has been developed to capture GPU data and includes their temperatures, powers, voltages and frequencies

- ***Out-of-band plugins:***

This family of plugins executes outside the calculation nodes:

- ipmi_pub: this plugin acquires data from sensors on the nodes' IPMI interface. These are mainly temperatures and power consumption.
- slurm_pub: it takes care of acquiring the data related to the job scheduler (Slurm) such as all the statistics about the jobs executed on the system

3.5.2 Submer API to get the information from the rack unit.

The following chapter explains the API call to get the live data from the Submer Rack unit.

Basepath

Description:

To access to the API endpoint, use:

http://{smartpodURL}/api

Where {smartpodURL} is the IP/URL address of the Rack unit inside your network

Example:

http://10.0.0.2/api

API call: GET /realTime

Description

Returns current live measurement values of the Rack unit

Return type
This call returns this model **inline_response_200**

Example data

The following payload represents an example of a return response.

Content-Type: application/json


```
{
  “data” : {
    “mpue” : 1.03,
    “cpu1temp” : 51,
    “demo” : false,
    « wf » : 51,
    « mode » : « normal »,
    « dissipationW » : 0,
    « cpu0temp » : 51,
    “temperature” : 45.7,
    “alarm” : 60,
    “cti” : 51,
    “pump2rpm” : 3000,
    “dissipation” : 2.02,
    “cto” : 45,
    “pump1status” : 1,
    “factory” : false,
    “cf” : 51,
    “setpoint” : 55,
    “test” : 0,
    “warnings” : [ {
      “warningType” : “LowCF”,
      “idWarning” : “1133”,
      “description” : “Low Coolant Flow”,
      “startTime” : “2022-07-20 18:14:59”
    }, {
      “warningType” : “LowCF”,
      “idWarning” : “1133”,
      “description” : “Low Coolant Flow”,
      “startTime” : “2022-07-20 18:14:59”
    } ],
    “dissipationC” : 0,
    “consumption” : 800,
    “wti” : 51,
    “fixedErrors” : [ “”, “” ],
    “fixedWarnings” : [ “”, “” ],
    “wto” : 51,
    “pump2status” : 1,
    “pump1rpm” : 3000,
    “maintenance” : 6,
    “errors” : [ {
```

```
        "idFailure" : "1133",
        "failureType" : "HighWT",
        "description" : "Hight Input Water Temperature",
        "startTime" : "2022-07-20 18:14:59"
      }, {
        "idFailure" : "1133",
        "failureType" : "HighWT",
        "description" : "Hight Input Water Temperature",
        "startTime" : "2022-07-20 18:14:59"
      } ]
    },
    "meta" : [ "meta", "meta" ]
    }
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- *application/json*

Responses

These are the supported response codes and their models.

200

Description: successful operation

Model: inline_response_200

Models

The models detail the each object with:

- Description
- Properties, with 33hreshold33n, property type, example

Failure

Description:

List of failures

Properties:

idFailure *String*

Unique identifier of the failure of the Rack unit

example: 1133

startTime *String*

Datetime (in GMT) when the failure has been detected

example: 2022-07-20 18:14:59

failureType *String*

Categorization code of the failure

example: HighWT

description *String*

Human readable explanation of the failure

example: Hight Input Water Temperature

FailureFixed

Description:

List of fixed failures

Properties:

idFailure *String*

Unique identifier of the failure of the smartpod

example: 1133

startTime *String*

Datetime (in GMT) when the failure has been detected

example: 2022-07-20 18:14:59

failureType *String*

Categorization code of the failure

example: HighWT

description *String*

Human readable explanation of the failure

example: Hight Input Water Temperature

endTime *String*

Datetime (in GMT) when the failure has been fixed

example: 2022-07-20 19:14:59

RealTimeMeasurement

Description:

Live measurerement of the Rack unit

Properties:

temperature *Float*

Current internal temperature (in Cº) of the tank

example: 45.7

cti *Float*

The current inlet temperature (in Cº) of the coolant in the PDU format: float

example: 51

cto *Float*

The current outlet temperature (in Cº) of the coolant in the PDU format: float

example: 45

cf *Float*

Current coolant flow (in m3/h) format: float

example: 51

wti *Float*

The current inlet temperature (in Cº) of the water in the PDU format: float

example: 51

wto *Float*

The current outlet temperature (in Cº) of the water in the PDU format: float

example: 51

wf *Float*

Current coolant flow (in m3/h) format: float

example: 51

consumption *Float*

Current Rack unit consumption (in W) format: float

example: 800

dissipation *Float*

Current dissipation (in kW) format: float

example: 2.02

mpue *Float*

Current mPue format: float

example: 1.03

errors *array[**Failure**]*

Current list of errors

fixedErrors *array[**FailureFixed**]*

Current list of fixed errors

warnings *array[**Warning**]*

Current list of warnings:

fixedWarnings *array[**WarningFixed**]*

Current list of fixed warnings

pump1status *Integer*

Indicates if the pump 1 is running (0->OFF; 1->Running)

Enum:

- *0*
- *1*

example: 1

pump1rpm *Integer*

Current speed of the pump 1 (in RPM)

example: 3000

pump2status *Integer*

Indicates if the pump 1 is running (0->OFF; 1->Running)

Enum:

- *0*
- *1*

example: 1

pump2rpm *Integer*

Current speed of the pump 1 (in RPM)

example: 3000

dissipationC *Float, format: Float*

Current dissipation of the coolant (in kW)

Note: Use dissipation property for dissipation monitorization.

Example: 0

dissipationW *Float, format: Float*

Current dissipation of the water (in kW)

Note: Use dissipation property for dissipation monitorization.

Example: 0

setpoint *Integer*

Current coolant temperature (in Cº) setpoint.

Note: this is a configuration value. Not a live measurement. format: int64

example: 55

alarm *Integer*

Current coolant alarm temperature (in Cº) 37hreshold.

Note: this is a configuration value. Not a live measurement.

Example: 60

cpu0temp

Float Current internal system temperature (in Cº) . Note: measurement not useful for tank behavior monitoring. format: float

example: 51

cpu1temp *Float, format: float*

Current internal system temperature (in Cº) .

Note: measurement not useful for tank behavior monitoring.

Example: 51

mode *String*

Current mode of the Rack unit

Note: measurement not useful for tank behavior monitoring.

Example: normal

test *Integer*

Indicates in the Rack unit is running a test routine

Note: measurement not useful for tank behavior monitoring.

Nullabe: Yes

maintenance *Integer*

Indicates if the Rack unit is running a maintenance routine

Note: measurement not useful for tank behavior monitoring.

demo *Boolean*

Indicates in the Rack unit is running a demo routine

Note: measurement not useful for tank behavior monitoring.

example: false

factory *Boolean*

Indicate if the Rack unit is in Factory mode Note: measurement not useful for tank behavior monitoring.

example: false

Warning

Description:
List of warnings

Properties:

idWarning *String*

Unique identifier of the warning of the Rack unit

example: 1133

startTime *String*

Datetime (in GMT) when the warning has been detected

example: 2022-07-20 18:14:59

warningType *String*

Categorization code of the warning

example: LowCF

description *String*

Human readable explanation of the warning

example: Low Coolant Flow

WarningFixed

Description:

List of fixed warnings

Properties:

idWarning

String Unique identifier of the warning of the smartpod

example: 1133

startTime *String, format: datetime*

Datetime (in GMT) when the warning has been detected

example: 2022-07-20 18:14:59

warningType *String*

Categorization code of the warning

example: LowCF

description *String*

Human readable explanation of the warning

example: Low Coolant Flow

endTime *String, format: datetime*

Datetime (in GMT) when the warning has been fixed

example: 2022-07-20 19:14:59

Inline_response_200

Description:
Response format of the request

Properties:

meta *array[String]*

Added information

Note: Not used for current context

data *RealTimeMeasurement*

Contains the current measurement

3.4.3 BSC HPC clusters specifications

Any metric value provided by the Rack unit will later be integrated to BSC HPC monitoring system tools, for further analysis.

Any BSC HPC clusters, there are 3 types of monitoring:

- Alert monitoring
- Performance monitoring
- Job post-mortem monitoring/reporting

Alert monitoring will follow the health status of any component and service of the supercomputer. For each failure, an email alert will be generated to the system administration or site infrastructure team in order to evaluate and react to the failure. For the implementation of this service, we propose the use of the open-source software Icinga.
Performance monitoring will monitor all performance and environment metrics related to any server. Performance monitoring system used is modular and is able to add metrics adapted to the different services provided by the supercomputers. BSC-CNS has huge expertise in designing and implementing in-house high scalable performance monitoring systems for the different clusters. Here is a generic list of metrics that will

be common to all servers (if available), based on the metrics collected right now in MareNostrum4:

- CPU metrics: load; CPU stat (user, idle, system, nice, wait)
- GPU metrics: load, power consumption
- Memory metrics: used, cached, free, total
- Network metrics (any interface): packets rx and tx, bytes rx and tx
- Filesystem metrics: IOPS per filesystem, type of OPs per filesystem (open, read, write), MB/s per filesystem read and write per server
- Environment: Temperature, power consumption per server

Specific metrics will be configured in those servers related to the services that they will be offering.

A web visualisation system has to provide graph capabilities to show all metrics stored in the system to permit the creation of dashboards that show performance status of any supercomputer. Software stack used to implement this performance monitoring system will be similar to the one being used right now in MareNostrum4:

- Collectd or Telegraf: Collecting metrics locally per each server and aggregating values at service node level
- InfluxDB: Time-series database to store all performance metrics
- Grafana: Data visualisation tool

Job post-mortem monitoring will be used to store all information of job finished in the batch scheduling system to a database so user support and system administration can apply Data Analytics procedures to infer knowledge from jobs executed at the supercomputer. This monitoring will be implemented using a SLURM plugin, initially developed at BSC-CNS, to push information of finished jobs to an ElasticSearch database. Apart from direct access to this database by other services or BSC-CNS staff team, a Kibana service is used to perform data and graph visualisation via a web interface.

3.6 BSC site survey

- Destination address: Plaça d'Eusebi Güell, 1-3, 08034 Barcelona
- Entrance restrictions: Truck limited by 8 tons.
    - Submer will take care about the shipping till the final location at BSC facilities
- Final location
    - BSC team is required to modify their facilities to allow the deployment of the Rack units of Submer according to the specifications detailed in 3.4 Section.
        - Lift Truck modification
    - Door access 240cm

The requirements for the installation of these racks at BSC facilities will need to include the following material and tasks:

- ***Electrical***

2 Units: provision and mounting of the unit of derivation for devices modular Schneider Canalis KSB160SM424 with 160 A 24 nodes, equipped with 2 protections triphasic of 4P with 50A 50KA and 1 triphasic protection of 4P with 16 A 50KA from Schneiders by each unit.

Figure 28: Graphic Electrical schedider with 160 A 24 nodes, equipped with 2 protections triphasic of 4P with 50A 50KA and 1 triphasic protection of 4P with 16 A 50KA from Schneiders by each unit

Provision and installation of 40m long hose free of halogen RZ1-K 0,6/1KV 5x10mm2. Including small material for installation and needed for supporting.

Figure 29: graphic picture 40m long hose free of halogen RZ1-K 0,6/1KV 5x10mm2

Provision and installation of 20m long hose free of halogen RZ1-K 0,6/1KV 5x2,5mm2. Including small material for installation and needed for supporting.

Figure 30: graphic picture 20m long hose free of halogen RZ1-K 0,6/1KV 5x2,5mm2

4 units; provision and installation of air Cetac female IP 65 of 63A 3F+N+T

Figure 31: graphic picture Cetac female IP 65 of 63A 3F+N+T

2 units; provision and installation of air Cetac female IP 65 de 16A 3F+N+T

Figure 32: graphic picture Cetac female IP 65 de 16A 3F+N+T

Cleaning of the affected area of installation

- ***Cooling***

Empty of the existing hydraulic installation.

Implementation of the 4 points in the hydraulic circuit for the impulsion and return of PPR 63mm in the existing water collector of PPR 200 diameter.

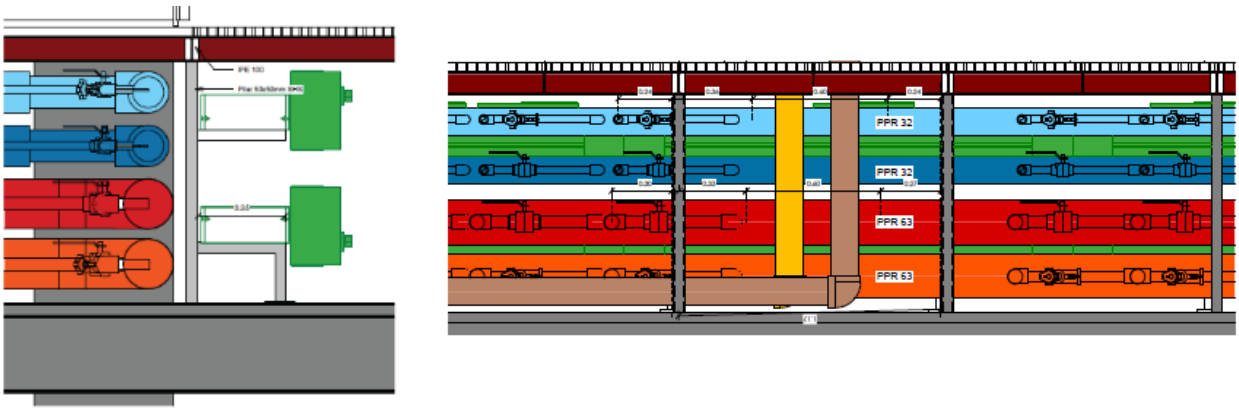

Figure 33: BSC existing hydraulic installation

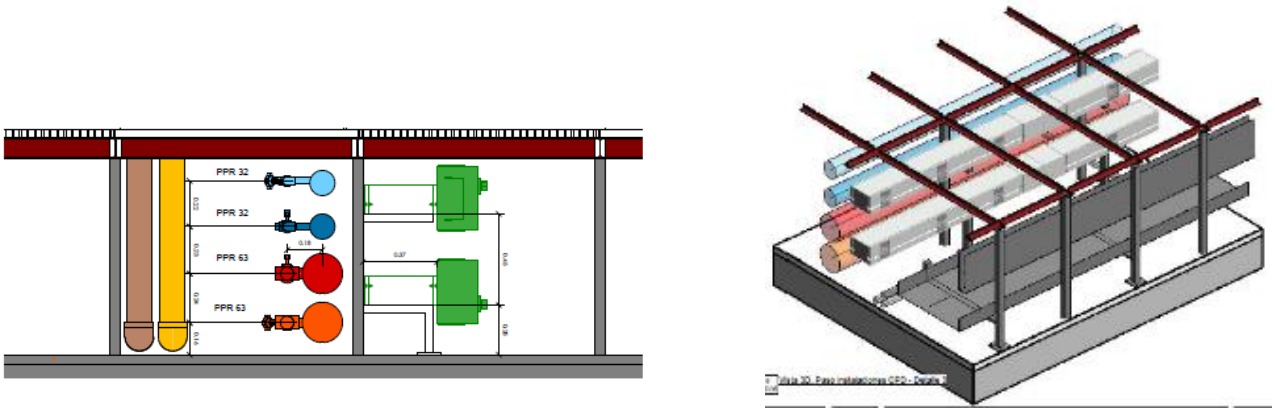

Figure 34: Figure 30: BSC existing hydraulic installation

4 units; provision and installation hydraulic kit Pettinaroli mod. XT 1692 1 ½" x 9000l/h. Including little material and parts for supporting.

Figure 35: hydraulic kit Pettinaroli mod. XT 1692 1 ½" x 9000l/h

4 units; provision and installation of the flexible hydraulic hoses of 1 ½" with INOX connections female screw in both extremes. It has to include male reduction from 1 ½" to 1 ¼" to connect to the Smart Pod and male 1 ½" at the infrastructure end.

Cleaning of the hydraulic circuit and fill of the circuit with osmosis water.

Fill Smart Pod hydraulic circuit, installation purged, regulation and hydraulic calibration.

Review of water pipes painting of the different circuits after the installation.

Cleaning of the affected area of installation

- ***Structure***

Metallic structure reinforcement installation for the raised floor and spreading of the weight for the Smart Pod. Per each Smart Pod, 4 units of IPN of 100mm x 60 cm long and 16 screws M12x50mm with nut and washer.

Cleaning of the affected area of installation.

4. Conclusion

The specifications Power density and scaling continue to be a challenge for the next generation exascale systems. The European PILOT will demonstrate advanced liquid immersion cooling technology required for the power densities resulting from increased scale, as well as integrated power modules and system management, supporting ultra-efficient rack power densities far surpassing today's solutions. The project plans to provide high-density immersion-compatible EAS accelerator system of 1OU compared to 3OU for commercial solution (i.e., Inspur, Supermicro, Gigabyte) as well as 0.33OU (1OU, 3N) Host system. The latter systems will be integrated and deployed into OCP compliant immersion cooling tanks that will enable a highly efficient and ultra-dense PILOT architecture capable of pushing rack densities up to 100kW and system efficiencies with a partial PUE of 1.03 or less.

5. Acronyms and Abbreviations

Acronyms and Abbreviations	
EAS	EUPILOT Accelerator system
MLS	Machine Learning Stencil
VEC	Vector
OAM	OCP-compliant Open Accelerator Module
UBB	Universal Base Board
HIB	Host Interface Board
OU	OpenU
LPDDR	Low Power Double Data Rate
C2C	chip-to-chip
CXL/PCIe	Compute Express Link
CPU	Central Processing Unit
OCP	Open computing Platform
CDU	Cooling Distribution Unit
MQTT	MQ Telemetry Transport
API	Application Programming Interface
GPU	Graphics Processing Unit