GreenDataNet

D2.3 – Server multi-level Software management

Final Version
EATON
EPFL
Rev 0.8
Contributors
EATON: Maria Laura Corallini, Gerald Guillaume, Dennis O'Sullivan, Arnaud Quette
EPFL: Pablo Garcia, Ali Pahlevan, David Atienza

TABLE OF CONTENTS

REVISION SHEET

Revision Number	Date	Brief summary of changes
Rev 0.1	27/07/2015	Baseline document
Rev 0.2	17/08/2015	EPFL updates
Rev 0.3	24/08/2015	EATON updates
Rev 0.4	26/08/2015	EPFL updates
Rev 0.5	27/08/2015	EATON modifications
Rev 0.6	28/08/2015	EPFL updates
Rev 0.7	31/08/2015	EPFL updated with experiments. Figures reworked.
Rev 0.7.1	01/09/2015	Fixed legend of the figures in the experiments.
Rev 0.8	02/09/2015	Final version

KEY REFERENCES AND SUPPORTING DOCUMENTATIONS

[1] EATON ePDU platform Brochure.
lit.powerware.com/ll_download.asp?file=Eaton%20EPDU-G3%20Tri-fold%20A4%20Brochure.pdf&ctry=80

[2] NUT – Network UPS Tools Website
http://www.networkupstools.org/

[3] MALAMUTE – White Paper
https://github.com/zeromq/malamute/blob/master/MALAMUTE.md

[4] The JavaScript Object Notation (JSON) Data Interchange Format, ISSN: 2070-1721

[5] libcurl - Multiprotocol file transfer library Website
http://curl.haxx.se/libcurl/

[6] RapidJSON - A fast JSON parser/generator for C++ with both SAX/DOM style API
https://github.com/miloyip/rapidjson

[7] NUT – Network UPS Tools Architecture
http://www.networkupstools.org/docs/developer-guide.chunked/ar01s02.html

[8] NUT – Network UPS Tools Architecture
http://www.networkupstools.org/docs/developer-guide.chunked/ar01s02.html

[9] "PUE Calculations: The Model and The Myths", White Paper by Sam Sheehan, CCG Facilities
http://www.ccgfacilities.com/pdf/CCG%20Whitepaper_%20PUE%20Calculations%20Model%20and%20Myths.pdf

[10] Redis Cookbook, by Tiago Macedo and Fred Oliveira, O´Reilly Media, 2011. ISBN: 978-1-4493-0504-8.

[11] "Guidance for Calculation of Efficiency (PUE) in Data Centers", White Paper by Victor Avelar

[12] J. Kim and et al., "Correlation-aware virtual machine allocation for energy-efficient datacenters," in Design, Automation & Test in Europe (DATE) Conference, 2013, pp. 1345–1350

[13] D. Dias and L. Costa, "Online traffic-aware virtual machine placement in data center networks," in Global Information Infrastructure and Networking Symposium (GIIS), 2012, Dec 2012, pp. 1–8

[13] Practical Applications in Digital Signal Processing 1st Edition, Richard Newbold, ISBN-13: 978-0133038385

2. INTRODUCTION

One of the main goals of the GreenDataNet project is to investigate and develop new technologies to decrease the power consumption of urban Data Centres and making them more sustainable in the future.

The contribution of this deliverable in the achievement of this goal is a powerful optimization tool that focuses on the minimization of the power consumption at the server level.

2.1 DOCUMENT PURPOSE

Deliverable *D2.2 – Analytical models for Data Centres*, presented a set of mathematical models describing the behaviour of the IT load (servers, storages, network switches) and how the evolution of this one has an impact on the cooling requirements of the Data Centres. The module developed by EPFL and presented in this document analyses the hardware performances of the IT equipment and maps it with the real server power consumption measurements collected by the Data Centre controller. The algorithm running inside this "EPFL module" studies the measurement collected and proposes corrective actions at the server level, such as switching the machine off, after migrating virtual machines from the current server to a more performing one, or changing the operational frequency of the machines.

For the first time, the optimization algorithm will be applied to real power consumption data collected by a rack controller which is able to communicate with the power devices installed in the DC such as intelligent Power Distribution Units (PDUs), Uninterruptible Power supplies (UPS) or the smart current sensors developed in D2.1. Previously, the algorithm used the power consumption information coming from the Hypervisor of the Server, which are imprecise as they are limited to the consumption of the chip and the tolerances of the IT embedded measurement system is higher than the one of the power devices monitoring system (Precision of 1% for the plug output measurement) [1].

The acquisition and monitoring module developed by EATON in this deliverable is structured according to the specifications provided in deliverable *D1.5 – Software architecture specifications,* this document focuses specifically on the monitoring of power systems

Therefore, the output of this deliverable is an evolution of D3.2. Firstly, the power consumption of the servers comes from real measurements, collected by a specific server developed for the monitoring of the Data Centre in order to achieve a higher level of measurement precision: from 10% of the power supply of the servers to 1% of the measurement in the distributed power unit.

2.2 DEFINITION, ACRONYMS AND ABBREVATIONS

2.2.1 KEY ACRONYMS AND ABBREVATIONS

AC	Alternating Current
AEMS	Aggregated Energy Management System
AMQP	Advanced Message Queuing Protocol Standard
BMS	Battery Management System
CER	Cooling Efficiency Ration
CIM	Common Information Model
CPU	Central Processing Unit
CUE	Carbon Usage Effectiveness
DC	Direct Current
DCIM	Data Centre Infrastructure Management
DoD	Depth of Discharge
DOW	Description of Work
EC	European Commission
EPA	Environmental Protection Agency
ERE	Energy Reuse Effectiveness
EV	Electric Vehicle
GC	GreenDataNet Controller
GHG	Greenhouse Gas
HVAC	Heating, Ventilation and Air Conditioning
IAB	Industry Advisory Board
ICT	Information and Communication Technology
IPR	Intellectual Property Rights
IT	Information Technology
ITEE	IT equipment Energy Efficiency
KPI	Key Performance Indicator
LEED	Leadership in Energy and Environmental Design
MoM	Message oriented Middleware
NIC	Network Interface Controller
NoC	Network-on-Chip
NUT	Network UPS Tools
OOB	Out Of Band (control channel)
PCC	Project Coordination Committee
PDU	Power Distribution Unit
ePDU	Rack Power Distribution Unit
PSB	Project Steering Board
PUE	Power Usage Effectiveness
PV	Photovoltaic
RC	Rack Controller, synonym of GC – GreenDataNet Controller
REF	Renewable Energy Factor
SDK	Software Development Kit

SEMS	Smart Energy Management System
SLA	Service Level Agreement
SME	Small or Medium size Enterprise
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SoC	System-on-Chip
SW	Software
UPS	Uninterruptible Power Supply
URI	Uniform Resource Identifier
WP	Work Package
WUE	Water Usage Effectiveness

2.3 DOCUMENT OVERVIEW

This document is divided in 5 main sections including the introduction. Section 3 describes the overall architecture of the Software, focusing on the interactions between the different main modules and the data exchanged and the communication interface.

Section 4 explains how the monitoring and acquisition layer works and how the measurements are collected from the power devices and sent to the management layer.

Section 5 details the structure of the management layer and describes the multi-level optimization algorithm.

3. SOFTWARE ARCHITECTURE

3.1 COMMUNICATION MODEL OF THE GLOBAL SYSTEM

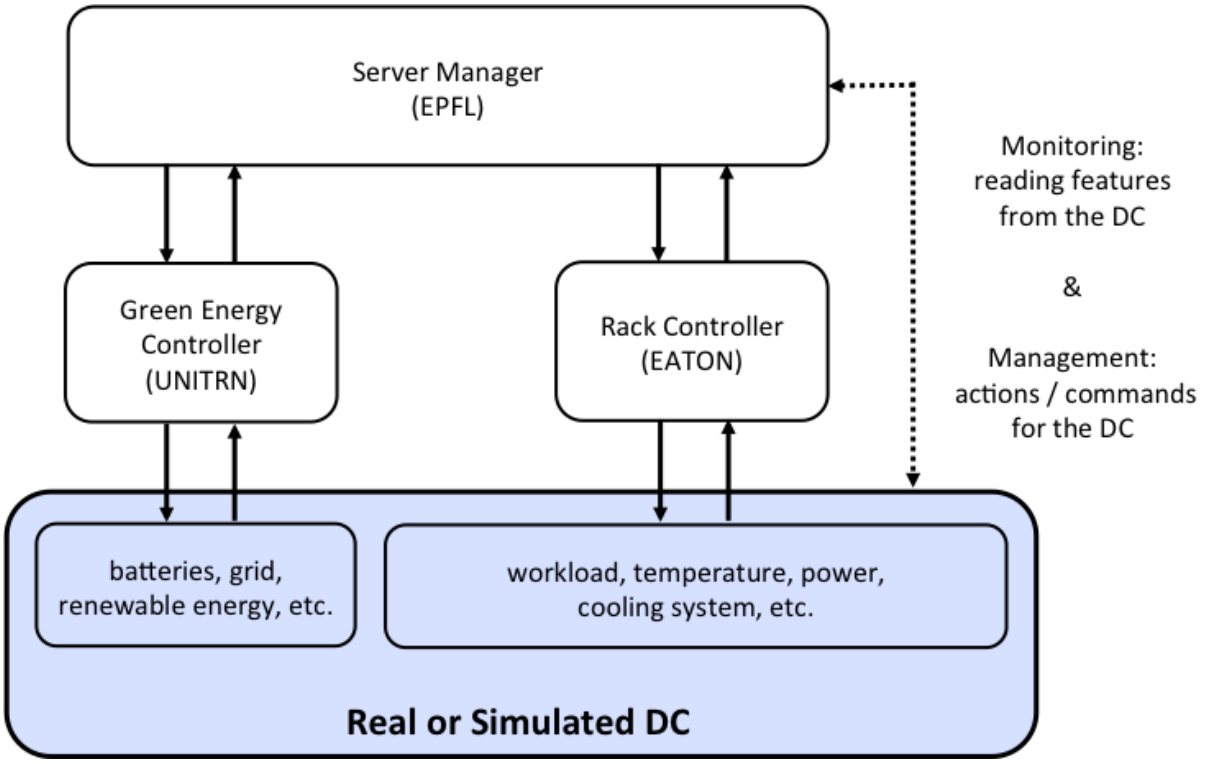


Figure 3-1 - Communication model overview

Figure 3-1 presents an overview of the communication model amongst the different modules that take part into the Server multi-level software management system; namely: the Server Manager, the Rack Controller, and the Green Energy Controller.

Details on the interaction EPFL – UNITRN were given in the previous deliverable D2.2, where we showed how the D2.5 (not yet delivered) and D3.2 interact together to achieve optimal VM allocation. Basically, the Server Manager had access to all the real-time data using the profiler of the tool (VMWare) that was in charge of the virtualization. Although this method offers a deep level of introspection, having access to the real direct measurements of power results into much more accurate calculations.

Therefore, and in parallel to this, Eaton was developing an API to access the data from their power distribution and management units, the Rack Controller, allowing a total control with accurate feedback including, for instance, the power losses due to inefficiencies, malfunctions, or incorrect setups. Both tools, the Server Manager and the Rack Controller have been conceived in a modular way, with the goal to make them Plug&Play with other, third party, tools. This deliverable, therefore, serves as an example of how such integration should be made.

Until this deliverable, the Server Manager was interacting directly with the DC (either the real one, or a simulated DC using artificial traces, see Figure 3.1): to get the power consumption, to trigger migrations, etc. Now, although this functionality is being integrated into the Rack Controller, there are still many features that are not supported (like reading the power per VM), for which we still have to interact directly from the Server

Manager to the DC. This is represented in Figure 3.1 as a dashed line. Eventually, all the functionality will be integrated into the Rack Controller and this channel will not be necessary anymore.

3.2 INTERACTION BETWEEN RACK CONTROLLER AND SERVER MANAGER

Using a client-server architecture facilitates the interaction between models. Both the SW from EATON and UNITRN run as services, check Figure **3-1**, launching a server that waits for requests in a predefined port. Then, the Server Manager starts the execution and periodically polls the other models for the information required.

3.2.1 STRUCTURE

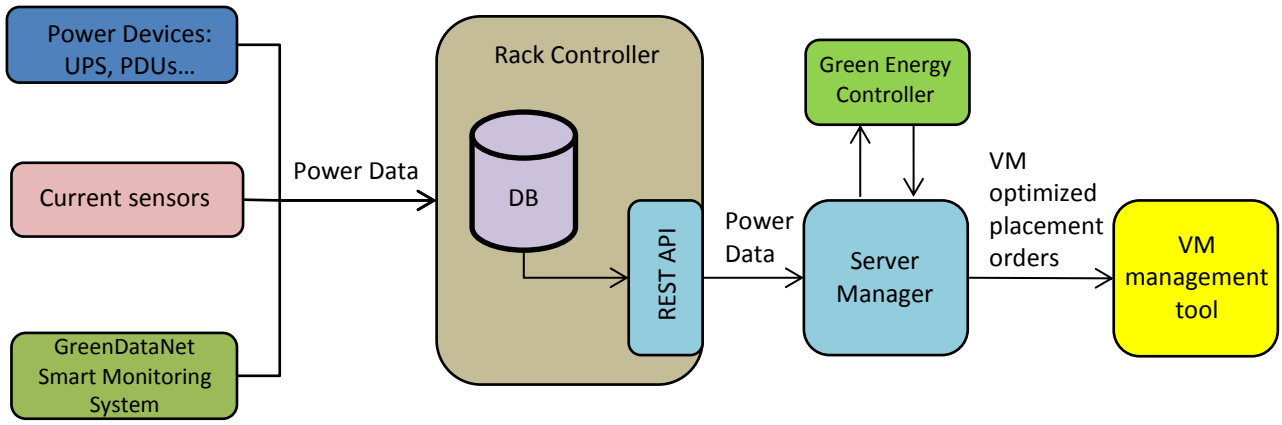


Figure 3-2 - Joint Software structure

The Rack Controller, from Eaton, provides control and monitoring features for the UPS, ePDU (Eaton´s PDU), and many other sensors in charge of monitoring the DC. All the information is stored in a database that can be accessed from Eaton´s API or directly through a local webserver, offering a uniform control and management interface. The development of the Rack Controller is done in C/C++, and based on several open source tools, including NUT [2]. It runs on a physical appliance called Rack Controller, on top of a Linux distribution for ARM processors.

The Server Manager is the algorithm that optimizes the allocation of VMs to servers. It is a headless SW, implemented in C++, that runs on x86 architectures (from regular computers to servers) as an additional task. In order to obtain the real-time information from the DC and send commands back to it, the Server Manager must interface with other modules, like the Green Energy Controller and the Rack Controller.

Next, we see in detail how both the Server Manager and the Rack Controller communicate during the DC optimization task.

3.2.2 INTERACTION BETWEEN MODULES

Deliverable D3.2 explains how the optimization system works using “time slots”, a way to discretize the timeline in “units of simulation”. At runtime, the Server Manager acquires the power consumption of the running components during the last time slot by polling the Rack Controller. This information is used to forecast the behaviour and to calculate the best allocation of VMs for the next time slot.

Note that, during the allocation phase, the Server Manager must also interact with the Green Energy Controller to achieve the best utilization of the renewable energies that may affect the VMs placement.

The Rack Controller is a modular system where additional components can be plugged-in into the SW system bus (through the Malamute agent, see ref. [3]) or interfaced through a REST API to provide more functionalities. While Malamute offers a tighter solution, it remains for now a local-only approach. The REST API provides a more seamless/flexible integration, covering remote access through the network, with the disadvantage that it is slower: requests in the order of 100ms; however, since this speed is enough for our current scenario, the REST API became our design choice. Data collection is done at the end of the time slots that can be configured from 5 seconds to minutes (shorter time slots result in higher accuracy).

In this first implementation phase, the Rack Controller exposes the power consumption of the plugs per ePDU. As explained in the example in paragraph 3.2.5, via the rest API it is possible to identify the entire power chain of a device; this means that, by knowing the power measurement of the plug to which the server is connected, we can access the power burnt per server.

3.2.3 COMMUNICATION INTERFACE

One of the services offered by the Rack Controller is a webserver that provides a REST API to external components. A client can then launch HTTP requests, and will receive the requested information encapsulated in JSON format [4]. JSON stands for JavaScript Object Notation, and it is a widespread lightweight data-interchange format.

The Server Manager has the role of the client in this communication; therefore, it has been enhanced with the capability to send and receive REST requests and parse and compose JSON packets, thanks to the libCURL [5] and RapidJson [6] libraries, respectively.

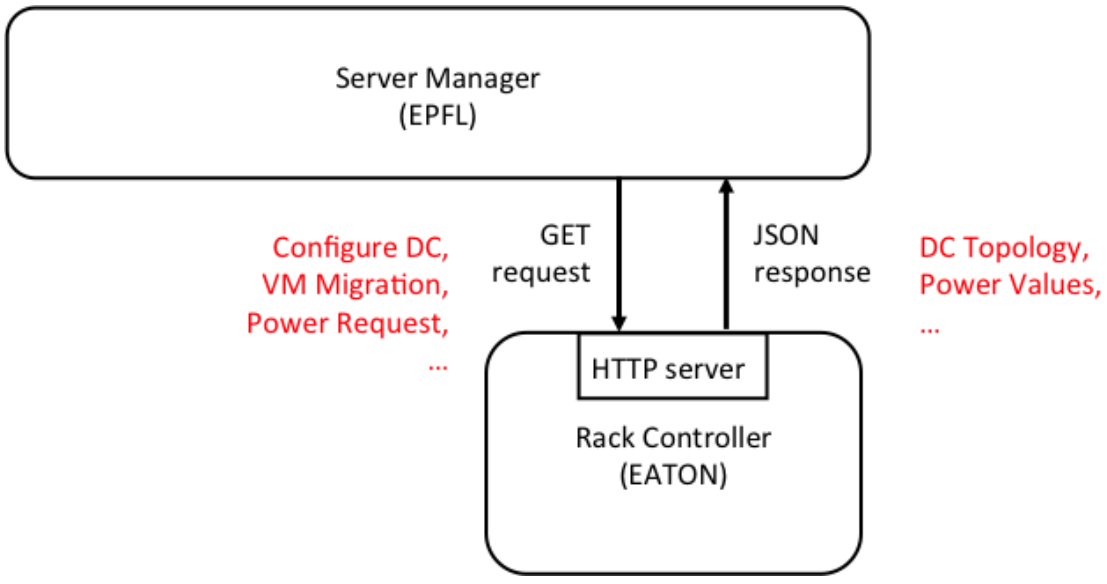


Figure 3-3 – Server Manager – Rack Controller communication

Figure 3-3 depicts the roles of the components, with the Rack Controller being the server, and the Server Manager acting as a client.

The JSON REST API is structured as follows:

- Only GET (read-only) requests are sent by the Server Manager to the Green Energy Controller
- The URI uses the format “http(s)://<rc_address>/api/vx/<category/<extra_param>” where :
    - http(s)://<rc_address> Rack Controller address and port.
    - /api REST API entry point.
    - /vx category version. For now, only “v1” is used.
    - /<category> application name or namespace:
        - “asset” to retrieve asset part of managed resources.
        - “metric” to retrieve metrics, indicators, pre-estimated graphic and detail historian measure.
    - /<extra_param> extra parameter depends on category.

The following list shows the main requests used by the Server Manager:

- Get the list of existing datacentres
  GET http://<host[:port]>/api/v1/asset/datacenters
- Get the topology of the datacentre
  GET http://<host[:port]>/api/v1/topology/location?from=DC1234&recursive=yes
- Get the power topology, to understand which devices power which servers
  GET http://<host[:port]>/api/v1/topology/power?to=<Server ID>
- Get the current values from a power device, to get the power consumption of a server
  GET http://<host[:port]>/api/v1/metric/current?dev=<PDU ID>

Next, we present some simple examples of accesses made through the API.

The logical way to determine the power chain of a server, from the Main Supply up to the plug to which it is connected, is first of all to define the physical position of the server; for example, if the server is installed in a cluster of Data Centres, it is necessary to know to which DC this device belongs to. By using two requests of the REST API it is possible to know the list of the DCs composing the cluster (Example 1, Paragraph 3.2.4) and the exact topology of each of them (Example 2, Paragraph 3.2.3): how many rooms and the corresponding name, how many rows of racks in each room and the devices installed in each rack.

Once the chosen device is selected, it is possible to get the complete power chain of a single device, so to know the exact path of the power supplying the server, including which plug of the Power Distribution Unit installed in the rack, supplies the server (Example 3, Paragraph 3.2.5). Once this connection is known, by getting the power measured by the plug, the corresponding power consumption of the server is known (Example 4, Paragraph 3.2.7).

3.2.4 EXAMPLE 1: LIST OF DATA CENTRES (DC)


```
{
  "datacenters": [
    {
      "id":"1",
      "name":"DC1"
    },
    {
      "id":"10",
      "name":"DC2"
    },
    {
      "id":"19",
      "name":"DC-LAB"
    },
    {
      "id":"35",
      "name":"DC-ROZ"
    }
  ]
}
```


3.2.5 EXAMPLE 2: A WHOLE RECURSIVE DC LOCATION TOPOLOGY REQUEST

The standard structure of a Data Centre is composed by rooms that are filled by racks disposed in rows, and the IT devices and ePDUs are installed inside the rack. It can happen, especially in small Data Centres that some of the devices may be installed out of this structure, so that they cannot be considered to belong to a specific row or room. The Figure 3.4 represent the structure of a small urban DC, composed of several kinds of elements (a room, a row, a rack and several devices):

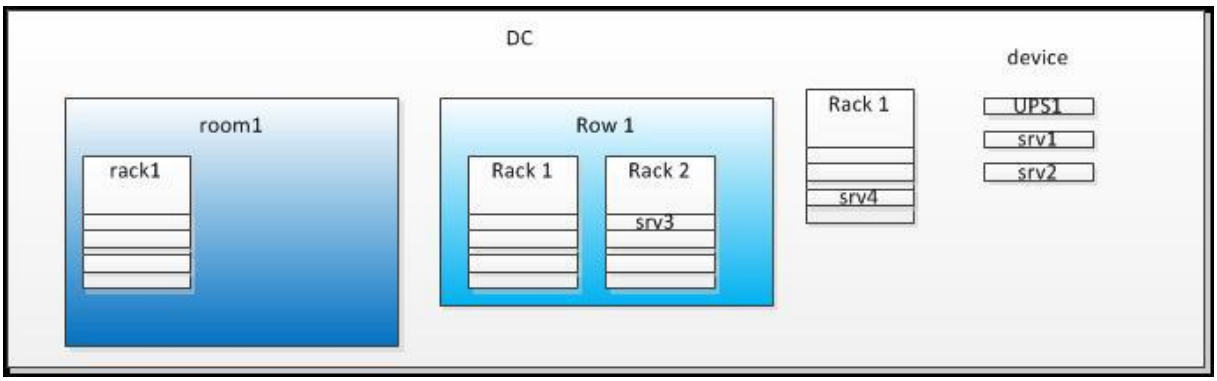


Figure 3-4 Random DC structure

GET /api/v1/topology/location?from=DC1234&recursive=yes
Response : HTTP 200


```
{
  {"name":"dc1","id" :"DC1234",
    "contains": {
      "rooms":[{"name":"room1","id":"RO1234",
        "contains":{"racks":[{"name":"rack1","id":"RCK12345"}]}
      }],
      "rows":[{"name":"row1","id":"RW1234",
```

```
          "contains":{
            "racks":[
                {"name":"Rack1","id":"RCK12346"},
                {"name":"Rack2","id":"RCK12347",
                  "contains":{"devices":[{"name":"srv3","id":"SRV1236","type":"server"}]}
                  }]}
        }],
        "racks":[{"name":"Rack1","id":"RCK1234"
          "contains":{"devices":[{"name","srv4","id":"SRV1237","type":"server"}]}
        }],
        "devices":[
          {"name":"UPS1","id" :"UP1234","type":"ups"},
          {"name":"srv1","id" :"SRV1234","type":"server"},
          {"name":"srv2","id" :"SRV1235","type":"server"}]
    }

}
```


3.2.6 EXAMPLE 3: POWER CHAIN OF A SERVER COMPOSED OF 1 FEED AND 1 EPDU

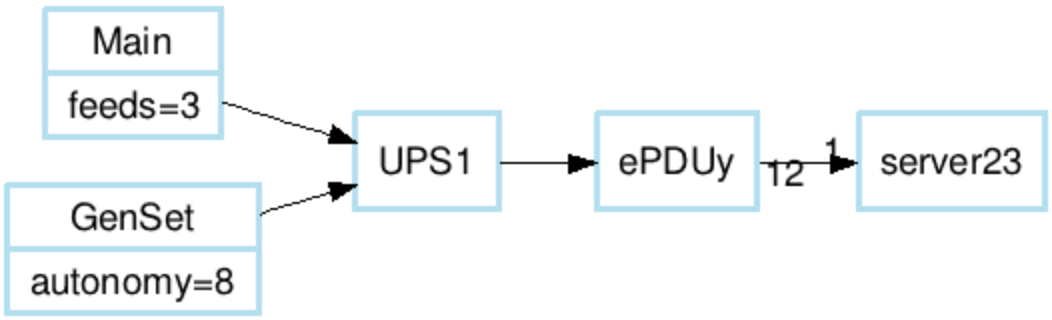


Figure 3-5 Server Power Chain

A power topology request for server23, whose ID is SVR1234, should be:

GET /api/v1/topology/power?to=SRV1234


```
{
  "devices":[
    {"name":"Main","id":"MA1234","type":"main"},
    {"name":"GenSet","id":"GS1234","type":"genset"},
    {"name":"UPS1","id":"UPS1234","type":"ups"},
    {"name":"ePDUy","id":"PDU1234","type":"pdu"},
    {"name":"server23","id":"SRV1234","type":"server"}],
  "powerchains":[
    {"src-id":"MA1234","dst-id":"UPS1234"},
    {"src-id":"GS1234","dst-id":"UPS1234"},
    {"src-id":"UPS1234","dst-id":"PDU1234"},
    {"src-id":"PDU1234","src-socket":12,"dst-id":"SRV1234","dst-socket":1}]
}
```


3.2.7 EXAMPLE 4: GET THE CURRENT VALUES OF AN EPDU

This is useful to get the power consumption of a server that is powered by a PDU.

GET /api/v1/metric/current?dev= PDU1234


```
{
   "current":[
      {"id": "PDU1234",
         "name":"pdu1",
         …
         "realpower.outlet.12":100}
         …
   ]
}
```

4. EATON RACK CONTROLLER

A Rack Controller is a mix of tightly integrated HW and SW, designed to largely leverage open-source technologies. It can be deployed in new data centres, as well as integrated into existing infrastructure. As such, it will have to interface with a number of HW and SW available to manage the DC infrastructure.

The rack controller is an IT appliance, whose system architecture is a combination of several components, or modules, running over a tailored and secured operating system. It runs on industrial hardware (physical appliance) or in a virtual machine (virtual appliance). This appliance is able to collect and treat the data coming from the devices of different nature, such as IT devices (servers, switches, storage), Power devices (UPS, PDU, ePDU) or sensors (smoke, current, temperature).

4.1 CORE MODULES

4.1.1 INTRODUCTION

Core modules are really the basic and mandatory modules to get a base GreenDataNet appliance working.
To understand what core modules are, it is important to present what the main data flow of GreenDataNet appliance is:

It is worth to note that:

- *Maintain RT values* also comprise data processing such as metrics calculation
- *Display* only emphasizes on graphical user interface
- *Action* comprises everything that is not simple data post-processing (such as metrics calculation) nor user interface display. Features such as the Server multi-level optimization belongs to this category.

4.1.2 DISCOVERY MODULE

This module is in charge of discovering the systems to be monitored. For now, the discovery is based on uploading a CSV formatted file, that describes the assets, including the Data Centre, and the devices that are part of it (servers, UPS, PDU, …). Once uploaded, the assets presence is verified, through a complementary discovery, to ensure that these are reachable. Afterward, the assets are inserted into the database, and the power monitoring module is configured for communicating with the devices.

4.1.3 INVENTORY AND ASSET MODULE

This module is in charge of collecting identification and processing topological data, to build topology maps (Basic power cabling status and Power chain description).
Asset module starts a new identification job in following situation:

- After Asset module initialization, checks all discovered, affected by master node and non-inventoried devices declared is its local Data Base Discovery table to do inventory.
- On equipment/RC affectation message reception from master node.

The asset module supports import feature for provisioning equipment and manages credentials of different users, this functionalities can be controlled through the REST API.

4.1.4 MONITORING POWER MODULES

These modules are part of the Collect stage presented in Paragraph 4.1.1 and they are in charge of:

- Collecting the devices data and health status (Key parameters and Critical alarms), through the communication interfaces described hereafter in the Protocols stacks chapter,
- Send collected data to real time data module.

The main monitoring system, used at least for power devices (UPS, PDU, …), is the NUT – Network UPS Tools. NUT is an Opensource project with a layered architecture:

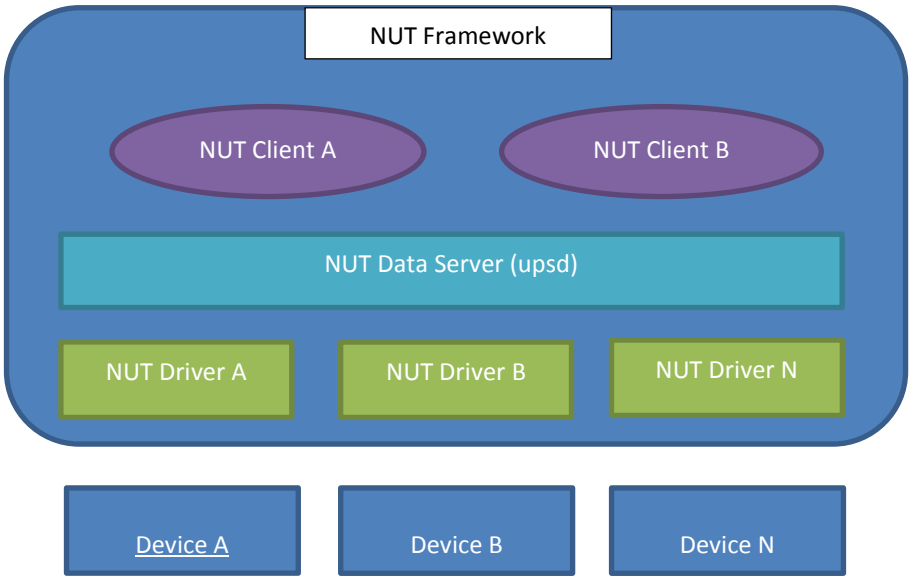


Figure 4-1 - NUT Framework architecture

Each NUT driver communicates with a device, transforming the data from a specific protocol (SNMP, USB/HID, …) into a common namespace. These data are then available to clients, through a network

protocol and a common API. Various command line tools and languages implementations are available.
Green Energy Controller embeds NUT, and its SW connects to it as a NUT client. Data from devices are then provided through the REST API.

For the simulated environment, the drivers used are a set of simulation drivers, mimicking the behaviour of real UPS´s and PDU´s. The Power Trace Synchronizer can then interact as a NUT client to request the modification of data inside the simulation drivers, and reproduce the actual impact resulting from the optimization algorithm.

4.1.5 REALTIME AND HISTORIAN STORAGE DATA MODULES

As explained in the introduction of the Core module in Paragraph 4.1.1, once instantaneous data are acquired , it will be treated firstly by the real time data module and then, after that the following slot of instantaneous data is acquired, it will be stored as an historian data. These operations are performed by 2 different modules:

- the real time data module
- the historian data module

The real time data module is composed of a data map where the live-data values are stored. Its goals are:

- Support real time response time GreenDataNet feature (between 1 and 2s max),
- Support notification mechanism, when a certain value changes beyond a given threshold,
- Aggregate data:
    - Data built from other data
    - Consolidated / compiled / reduced data
- Send events and alerts, according to status and configuration
    - Through MoM, for GreenDataNet dashboard and other GreenDataNet plugins
    - Through syslog, REST interface and SNMP traps, for 3rd Party Systems integration
- Implement current value request API.
- The data map is allowed to grow as the number of tags increase.

The second part is the historian data module. Its goals are:

- Encapsulate the data and store it in persistent way. Filters like Deadband [???] may be available to store complete or partial values,
- Replicate (mirror) data on other storage node to get redundancy, ease of use, availability (including resilience) or performance,
- Implement historian value request API,
- Aggregate data (Compute trends),
- Archive old data and remove them from the on-line historian DB.

When replication is activated, the historian module pushes modified data on a specific topic which is routed to other RC. In this version, only RC-master is notified.

With this architecture the live data and the static data are separated.
Each other module would send messages to the storage module to read and/or write data.
Both the real time and historian module are implemented through a relational database, using the open source project MySQL.

4.2 POWER MODULES

4.2.1 POWER MEASUREMENTS POINTS ALONG THE DATA CENTRE POWER CHAIN

To establish accurate efficiency metric calculation figures it is essential that a clear and logical strategy is implemented to measure the power in key areas within the Data Centre infrastructure. The following are four such areas that should be measured:

1. Sub-meters - A clear indicator of the actual power to the Data Centre, as opposed to the overall power figure shared with building facilities such as offices, loading docks, equipment assembly rooms etc.
2. UPS – Output power provided to critical IT equipment.
3. Rack – Metered rack power strips/tap boxes or wall mounted RPP representing the aggregate to the rack.
4. Individual rack outlets – Use of Intelligent Power Strips (ePDU's) to monitor the power consumption of the equipment.

Continuous measurements are a clearly defined strategy to monitor the power consumption, which can be undertaken periodically or by continuous instrumentation. The policy of on-going measuring will provide alerts to unexpected inefficiencies but also confirmation of any improvements that might be made to the Data Centre.

Identifying areas where the Data Centre Manager is lacking effective means of monitoring their site allows to make the necessary changes to have full monitoring and metric measuring capability within the Data Centre. By understanding the Data Centre performances and its limitations it is possible to implement infrastructural upgrades that mitigate ineffective monitoring and management capabilities and improve the quality of information in relation to their site.

As we move to a larger scale Data Centre, the level of managed and measurable equipment greatly increases. Given the level of investment in this equipment there is not only a wish for more accurate and detailed measurements along the power chain, there is an absolute requirement to understand how power is being consumed and where any inefficiencies might be occurring.

Below is purely an example of how power may very well be consumed by the different elements within the Data Centre:

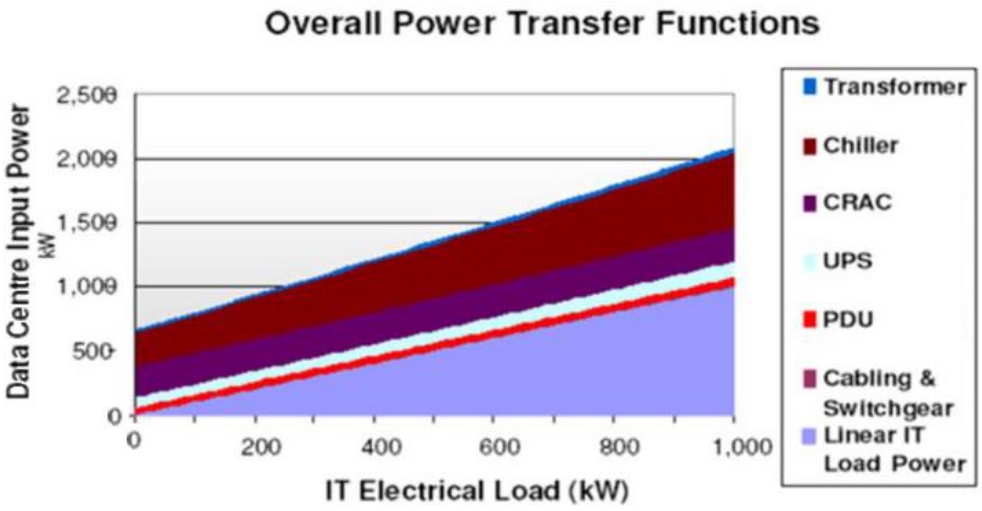


Figure 4-2 - DC Power consumption split

Without measurement capabilities it would be impossible to identify each point of consumption: It is important to be able to identify the levels of power consumed by each element in the Data Centre power chain because, as Figure 4.2 shows, even if we remove the IT electrical load, the Data Centre still uses significant power.

The key to monitoring will always revolve around strategic uptime capabilities. By proactively monitoring environmental variables, Data Centre managers are able to greatly reduce their risk of having extended downtime. This, in turn, creates a more robust and easier to manage Data Centre. With this in mind the ability to calculate an accurate measurement greatly depends on the points at which the measurement is taken. Before any realistic power management strategy can be implemented it is essential to fully understand how the power consumption is split across the physical infrastructure; simply put: first understand, then, manage. Outlined in the image below are the points along the power chain at which it is advisable to measure and the level of detail for each measurement:

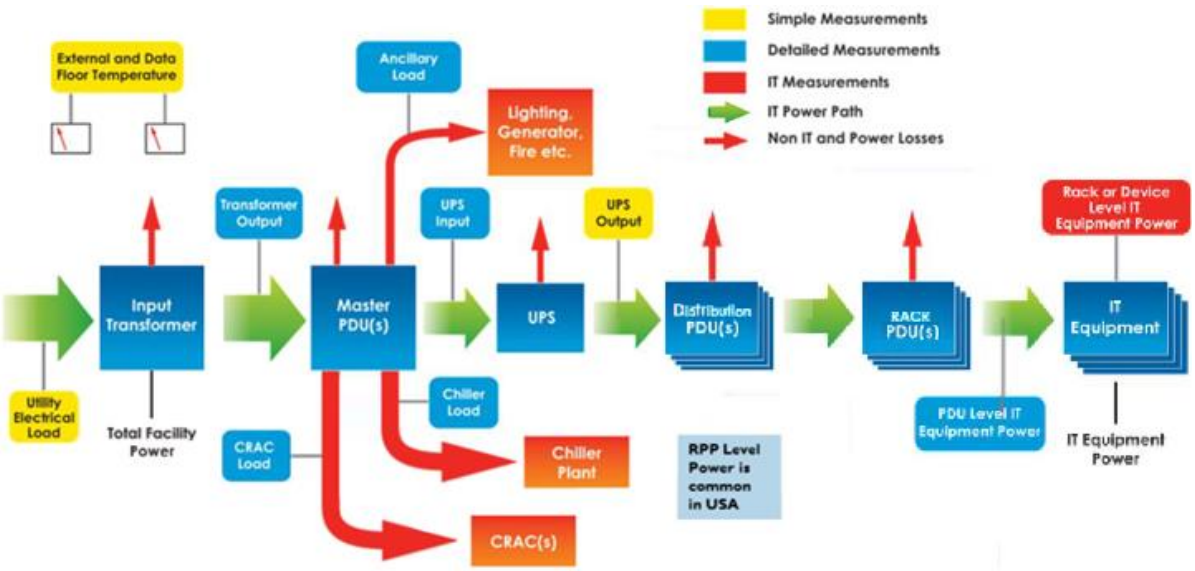


Figure 4-3 Power chain losses

One definite area of confusion to avoid would be to solely rely on measurements at the furthest end of the power chain such as the Rack mounted PDU and IT monitoring e.g. IPMI. There is a definite value to these measurements as they are displaying the final steps the power takes along the chain. However, as they are at the lower end of the power chain they do not allow for the display of any power losses ranging from the higher end down.

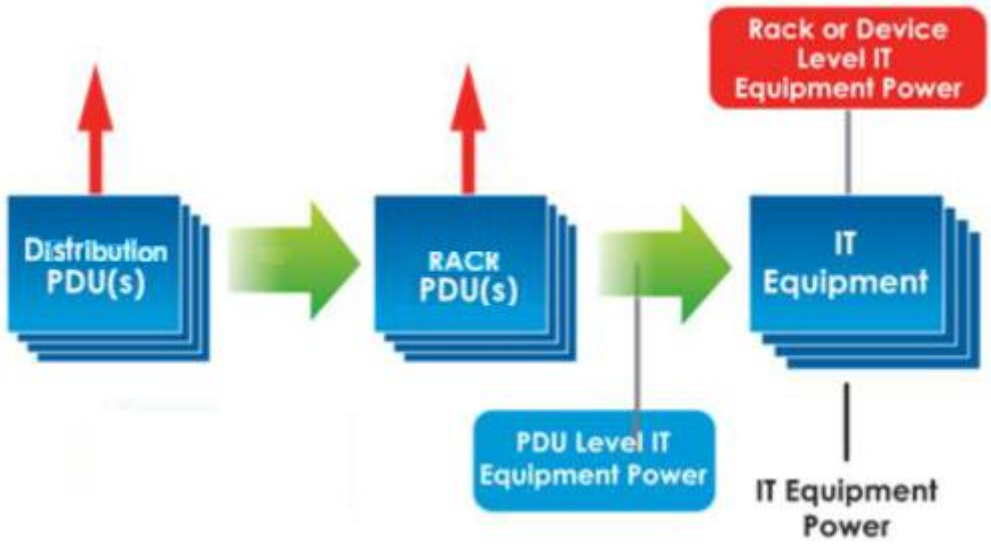
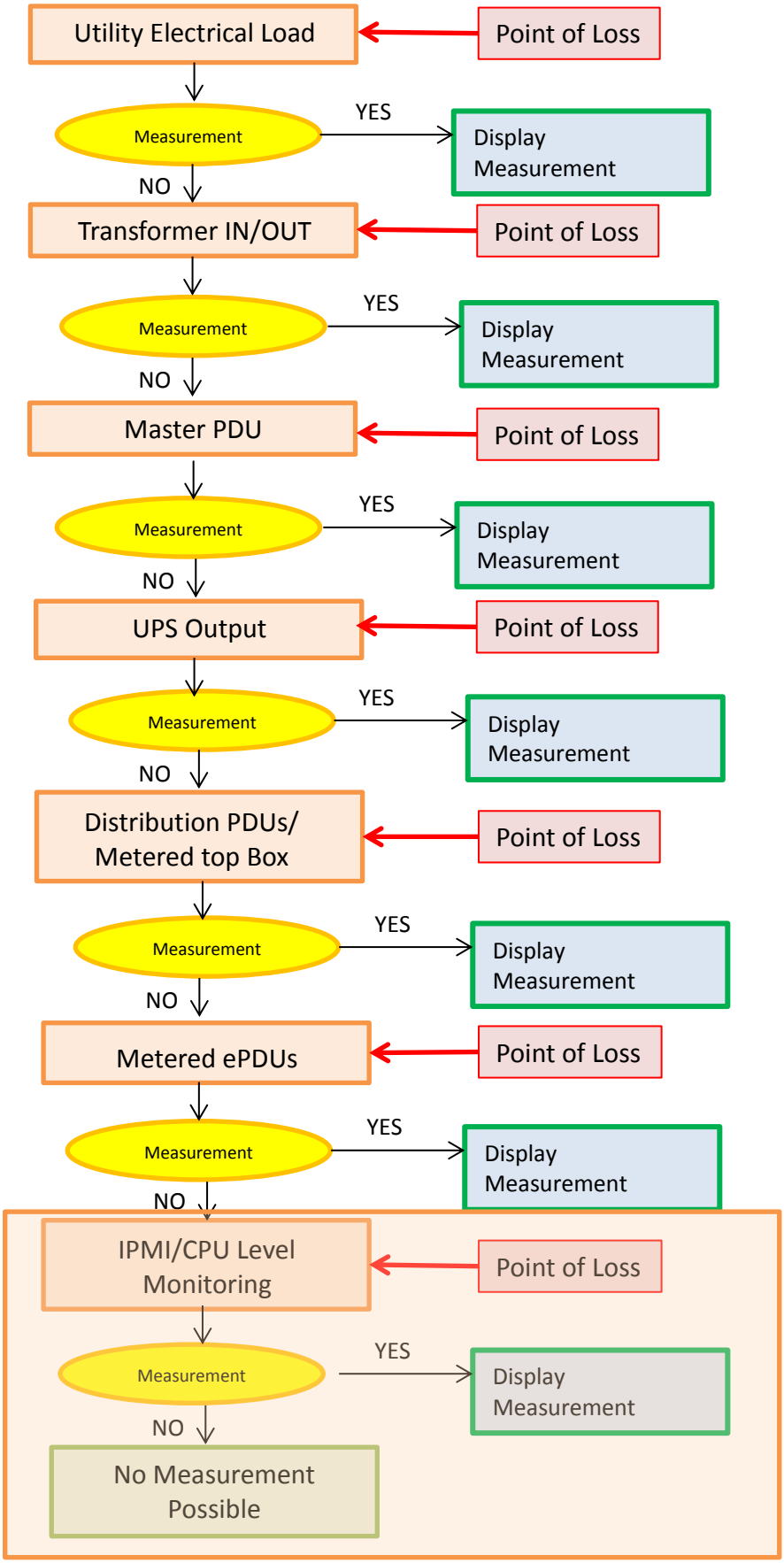


Figure 4-4 - Functional representation of a DC Power chain structure

The flow chart in Figure 4.5 is a representation of a simple power chain, not only does it show the potential points of measurement, it also shows the points at which power loss can occur. By actively monitoring the entire power chain, it is possible to easily identify each point of loss:



Not considered because of the lack of precision of the measurement system

Figure 4-5 List of measurement and loss points

The main focus of Data Centre in today's market is to be able to prove the efficiency of the facility. The most commonly used efficiency performance metric within the Data Centre industry is PUE (Power Usage Effectiveness). As mentioned before, the resulting data varies depending on the point at which the measurement was taken, which can also impact on the overall PUE result. To that end the Data Centre industry generally recognizes 4 categories of PUE. All four PUE Categories require that the Total Data Centre Facility Energy reflect all energy sources serving both the Data Centre and other spaces. However, the categories quantify IT energy quite differently with respect to the location of measurement. The table below provides a summary of energy elements used in the four PUE category calculations recognized by industry:

PUE ENERGY ELEMENTS				
	PUE Category 0	**PUE Category 1**	**PUE Category 2**	**PUE Category 3**
IT Energy	• Instantaneous peak kW demand • Measured at UPS output	• Total annualized kWh consumption • Measured at UPS output	• Total annualized kWh consumption • Measured at PDU output	• Total annualized kWh consumption • Measured at IT input
Total Data Centre Facility Energy	• Instantaneous peak kW demand • Measured at facility input	• Total annualized kWh consumption • Measured at facility input	• Total annualized kWh consumption • Measured at facility input	• Total annualized kWh consumption • Measured at facility input

Table 4.1 - PUE category and corresponding measurements [8]

4.3 COMMUNICATION PROTOCOLS

4.3.1 COMMUNICATION TO SINGLE POWER DEVICEAND MACHINE TO MACHINE

The Rack Controller is able to communicate with the other devices through the following protocols:

- SNMP
- Eaton XML/PDC (Power Device Class)
- Extendable to all the protocols supported by NUT [9]

For interaction with other Software's, as explained before with the description of the interaction with EPFL Software, is the REST API.

5. EPFL SERVER MANAGER

In this section, we present the Server Manager, a multi-level and multi-objective framework for the optimization of green virtualized DCs, to jointly minimize the energy consumption and the carbon footprint, exploiting renewable energy sources, VMs allocation schemes and electric systems.

5.1 ALGORITHM

The optimal solution to the problem of VM placement in a set of DCs is of NP-complexity and, therefore, unsuitable for real-time scenarios. The decision to place a VM in a certain server affects all the parameters of the DC: from the speed of the computations to the lifetime of the battery. In order to tackle this problem, we have decomposed the optimization problem into two sub-problems: VM clustering for each DC and VMs allocation to server-rack mapping.

Compared to state-of-the-art solutions, we take a holistic approach and consider a few additional metrics. Based on data and CPU-load correlation properties, for instance, two cost functions are defined to calculate the forces of attraction and repulsion such that highly data-correlated VMs should be clustered together, while highly CPU-load correlated VMs should be placed apart.

5.2 STRUCTURE

For each DC, we consider a capacity cap (in Joules) computed according to the available battery energy, renewable energy forecast and power consumed in the DCs during the last previous time slot. Then, the VMs are clustered for the DCs according to their energy caps and VMs loads. We utilize a modified version of the k-means algorithm to cluster the VMs with respect to each cluster (DC) capacity cap, and the distance between two VMs obtained from the repulsion and attraction phase in the 2D plane. In this step, we do not consider network latency for the migrations as a QoS criterion to obtain the optimal solution. At a later step, the k-means output is revised to meet the hard time constraint for migrating VMs across DCs.

At local level, each VM is allocated to a server, and the optimal Voltage/Frequency operating for all the servers is computed. In this step, we consider only CPU-load correlation to allocate VMs to the minimum number of servers. We utilize the best state-of-the-art algorithm [10] in this phase. The CPU correlation-aware VM allocation method has been proposed to efficiently compact more VMs (in terms of CPU Million Instructions per Second (MIPS)) to the lowest number of servers across a certain time horizon. Finally, an optimal voltage/frequency (V/f) level is provided to achieve power savings without any QoS degradation. The VMs are allocated such that the CPU correlation among the allocated VMs in the server is minimized, and the number of the active servers is minimized while the server does not exceed its total CPU capability and satisfying performance requirements. This correlation-aware VM allocation algorithm is periodically invoked at every time-slot.

After allocating all VMs to the servers in all the DCs at time slot T, we utilize the Green Controller inside each DC to manage energy sources during the time interval of [T; T + 1] based on the real provided renewable energy and energy consumption of the DC to minimize the cost while deciding to charge or discharge the battery based on the current electricity price.

6. RESULTS

In order to validate the integration of the Server Manager and the Rack Controller (see Figure 3-3), we have run a complete use case to test that all the SW is working correctly and both components can interact in real time.

6.1 EXPERIMENTAL SETUP

As the experimental setup, we have defined a dynamic system consisting of 50 VMs, on average, running on 1 DC with 1 room containing 2 racks with 10 homogeneous servers each. The server configuration is an HP ProLiant DL160 G6 server with 6 cores and two frequency levels (2.0GHz and 2.3GHz). Apart from the IT, the size of PV plant and battery (lithium-ion battery bank with 50% of DOD) are 5 kWp and 9600 Wh., and we have considered the two-level energy price scenario typical in Europe.

In order to simulate a realistic scenario, in terms of VMs and energy demand, we sample the CPU utilization of a real DC setup every 5 minutes for one day, then we replicate the samples up to 7 days with different time shift. Finally, to generate different samples for each day, we synthesize fine-grained samples per 5 seconds with a log-normal random number generator, whose mean is the same as the collected value for the corresponding 5-minutes sample rate. Arrival and life-time of each VM, given in time slots, are randomly generated by poisson and exponential distributions, respectively.

Finally, the DC energy controller is invoked every one hour, and the green online controller is invoked every 5 seconds.

6.2 EXPERIMENTAL RESULTS

In order to evaluate the joint schemes, i.e. VM allocation, green controller and monitoring system, I demonstrate three metrics: energy consumption, operational cost and the amount of solar energy usage in the time horizon of one week.

Figure 6.1 shows the energy consumed, including IT equipment and cooling system, every hour, by the DC for one week. The total energy consumption is 285 MJ. As explained in Section 5, the clustering algorithm favors consolidation and leads to power savings by lowering the number of active severs in the DC based on the CPU-load correlation of the VMs.

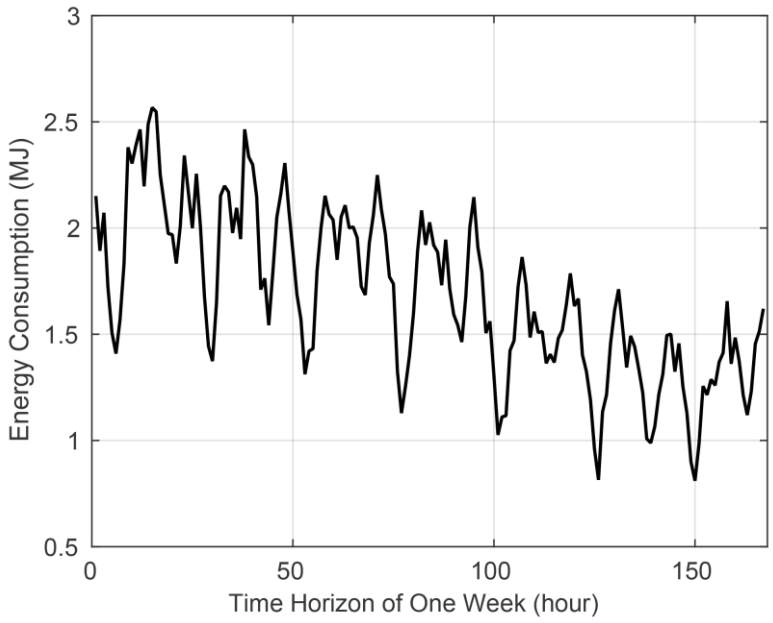


Figure 6-1 - DC Energy consumption

Next, Figure 6-2 provides better insight, reporting the computing energy consumed per each rack in the same time horizon of one week. For energy efficiency, the VMs are run in the minimum number of servers and, accordingly, in the minimum number of racks. Therefore, the allocator first fills up Rack1, using only Rack2 when more computing power is required. As a consequence, Rack2 consumes less energy compared to Rack1.

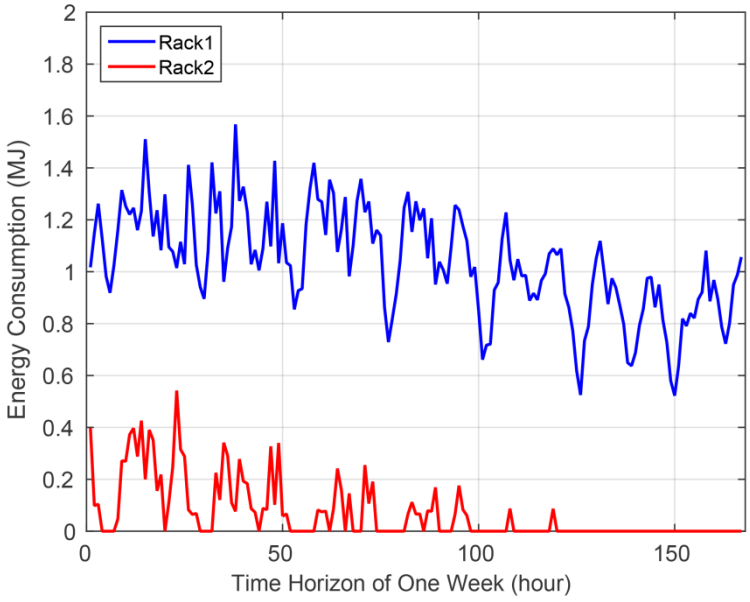


Figure 6-2 – Energy consumption per rack

Within the Server Manager, the DC energy sources are used efficiently based on the grid price and available renewable and battery energy. Figure 6-3 shows the normalized operational cost exhibited by the DC in the time horizon of one week, and Figure 6-4 depicts the usage of the

renewable energy during the same period. The Green Controller optimizes the amount of renewable energy used for the DC load and for the battery charging, to reduce the DC dependency on grid energy.

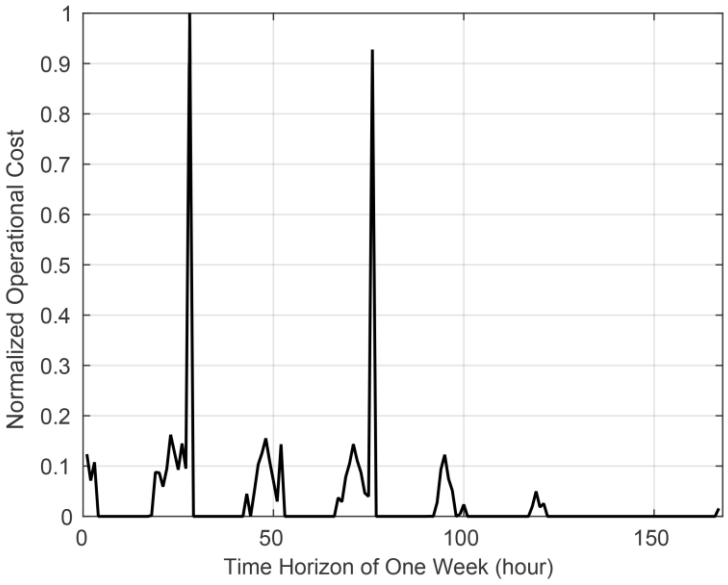


Figure 6-3 – DC Operational Cost

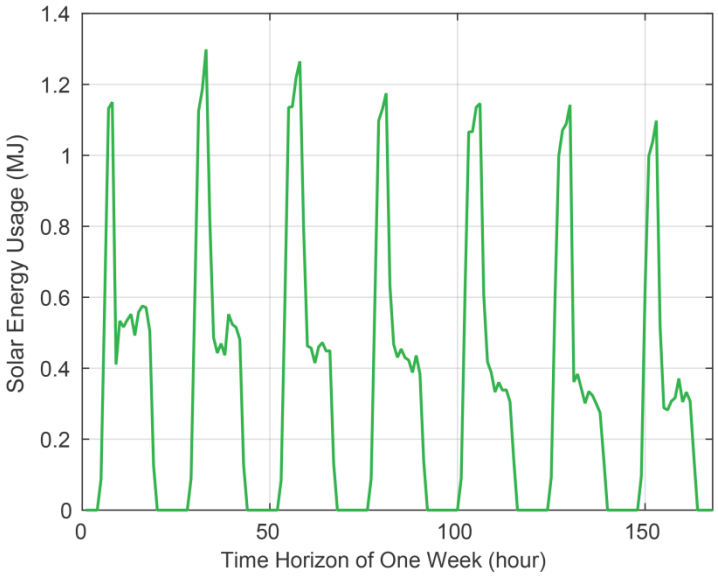


Figure 6-4 – DC renewable energy usage

7. CONCLUSIONS

With the upgrade presented in this deliverable, the optimization algorithm for VM allocation can now communicate with the power devices installed in the DC (such as intelligent Power Distribution Units (PDUs), Uninterruptible Power supplies (UPS) or the smart current sensors developed in D2.1), through a dedicated module that provides a clean interface: the Rack Controller. These readings provide information about the complete power chain, as opposed to the initial system (see D3.2) that was aware only of the final consumption at the server level. Once the full functionality is added to the Rack Controller, goal of D2.4, the algorithm will provide more accurate results, and will allow, for instance, better tracking of the power losses and failures.

As observed in the experimental test, the algorithm for VM allocation studies the measurements collected, and triggers actions at the server level (switching the servers off, scaling the frequency, etc.) in order to achieve the preset goals. D2.4 will focus on the optimizations that can be made at the Rack level.