Funded by
the European Union

TOWARD FULL IMPLEMENTATION OF THE KM3NeT RESEARCH INFRASTRUCTURE

KM3NeT – INFRADEV 2 – HORIZON – 101079679

Report on the proposed transformation to Green Computing

KM3NeT-INFRADEV2 GRANT AGREEMENT DELIVERABLE: D5.2	
Document identifier:	KM3NeT-INFRADEV2-WP5-D5.2-vfinal
Date:	31/12/2024
Work package:	WP5
Lead partner:	NCSR-D
Document status:	FINAL
Dissemination level:	PUBLIC
Document link:	https://www.km3net.org/km3net-eu-projects/km3net-infradev2/infradev2-outputs/

ABSTRACT

Physics collaborations employ computing centers for the processing and the analysis of large datasets that come from the experimental apparatuses. This need requires significant amounts of energy, and electric power, and results in the emission of vast amounts of carbon dioxide in the environment. The goal of this working package was to devise methods and procedures that can help to reduce the carbon imprint of data centers. Several ideas were considered focusing on what can be done at the hardware and software level separately. Additionally, an energy efficient data processing scheme is being proposed. We hope that the conclusions of this study can contribute and help the Kilometer Cube Neutrino Telescope (KM3NeT) collaboration to minimize the carbon emissions that are related to its computing resources.

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency (REA). Neither the European Union nor the REA can be held responsible for them.

I. COPYRIGHT NOTICE

II. DELIVERY SLIP

	Name	Partner and WP	Date
From	Leonidas Kalousis	NCSR-D, WP5	
Author(s)	Leonidas Kalousis	NCSR-D, WP5	03/12/2024
Reviewed by	Jutta Schnabel Christos Markou	FAU, WP4 NCSR-D, WP5	30/12/2024 30/12/2014
Approved by	Paschal Coyle KM3NeT IB	CNRS, WP1	31/12/2024

III. DOCUMENT LOG

Issue	Date	Comment	Author/Partner
1	03/12/2024	1st version	Leonidas Kalousis, NCSR-D
2	16/12/2024	First review	Victoria Ciarlet, CNRS
3	31/12/2014	Final version	Leonidas Kalousis, NCSR-D Victoria Ciarlet, CNRS

IV. APPLICATION AREA

This document is a formal deliverable of the Grant Agreement of the project, applicable to all members of the KM3NeT – INFRADEV2 project, beneficiaries and third parties, as well as its collaborating projects.

V. TERMINOLOGY

GHG	Greenhouse gas
CO_2	Carbon dioxide
KM3NeT	Cubic Kilometer (KM3) Neutrino Telescope
WP	Work Package
WP5	Work Package 5
PDU	Power Distribution Units
DCIM	Data Center Infrastructure Management
CRAC	Control room air conditioning
OS	Operating system
APP	Application
IT	Information and Technology
DAQ	Data Acquisition
TOT	Time over threshold
DST	Data Summary Tree
CM	Collaboration Meeting
MC	Monte Carlo

VI. LIST OF FIGURES

VII. PROJECT SUMMARY

The Kilometer Cube Neutrino Telescope (KM3NeT) is a large Research Infrastructure (RI) comprising a network of deep-sea neutrino telescopes in the Mediterranean Sea with user ports for Earth and sea science instrumentation. During the EU-funded Design Study (2006-2010) and Preparatory Phase (2008-2012), a cost-effective technology was developed, deep-sea sites were selected and the Collaboration was formed in 2013. This proposal constitutes a second INFRADEV project dedicated to KM3NeT in order to implement an efficient framework for mass production of KM3NeT components, accelerate completion of its construction and provide a sustainable solution for the operation of the RI during ten or more years. Following the appearance of KM3NeT on the 2016 ESFRI Roadmap and in line with the recommendations of the Assessment Expert Group, this project addresses the Coordination and Support Actions (CSA) to prepare a legal entity for KM3NeT, accelerate its implementation, establish open access to the RI and its data and ensure its sustainability by implementing an environment-friendly operation mode and evaluating the Collaboration socio-economic impact.

VIII. EXECUTIVE SUMMARY

This report summarizes the results of the sustainability studies associated with the computing resources of the KM3NeT collaboration. The study was conducted in the context of the WP5 of the KM3NeT-INFRADEV2 project. It consists (mainly) of two parts. In the first part we consider proposals that can be implemented at the hardware level and reduce energy consumption and carbon dioxide emissions. In the second, we examine software solutions that can be employed towards the same objective. Additionally, we also present a data processing scheme that appears to be energy efficient and environmentally friendly.

IX. TABLE OF CONTENTS

1. Introduction

It is widely accepted that global warming, and the associated climate change, is primarily caused by us, humans. The emission of greenhouse gasses (GHGs) through the combustion of fossil fuels has been identified as the main culprit [1]. GHGs are transparent to sunlight but they do absorb the heat emitted by the earth raising thus the temperature of the atmosphere and the planet. On the other hand, the burning of fuels is widely used for the production of energy. In this sense, one might say that current energy production is a driver for climate change.

There are approximately eight thousand data centers around the globe. Fig. 1 shows the major countries with the most data centers. They consume a total of roughly 196 - 400 TWh annually [2]. This amount corresponds to an emission of GHGs, especially that of carbon dioxide (CO_2), at a level comparable to the whole aviation industry. The purpose of work package 5 (WP5) was to identify solutions that minimize the CO_2 imprint associated with the computing resources of the Cubic Kilometer Neutrino Telescope (KM3NeT) collaboration. For this purpose, both hardware and software solutions have been considered. The content of this subject falls in the category that is commonly known as **sustainable computing** or **green computing**.

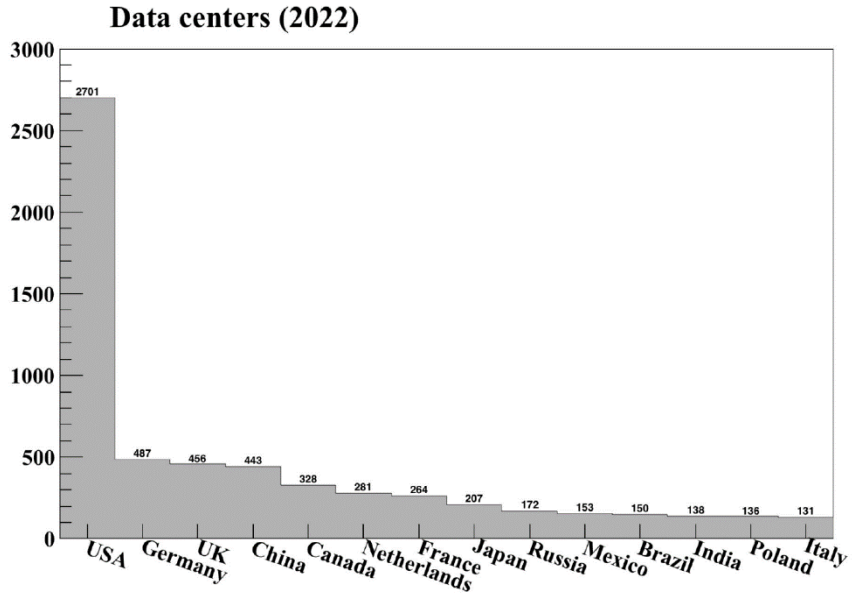


Figure 1 : *List of major countries with data centers.*

The report is organized as follows. In Sec. 2 we present solutions that can be implemented at the hardware level in order to reduce GHG emissions. Software ideas are then

treated in Sec. 3, while in Sec. 4 we put forward a strategy for data processing that has been designed to be environmentally friendly. In section 5 present what individual users can do towards this objective. Section 6 deals with the possibility to power up data centers using renewable sources of energy. Finally, we close this note summarizing the main results of this study and outline the future steps of this effort.

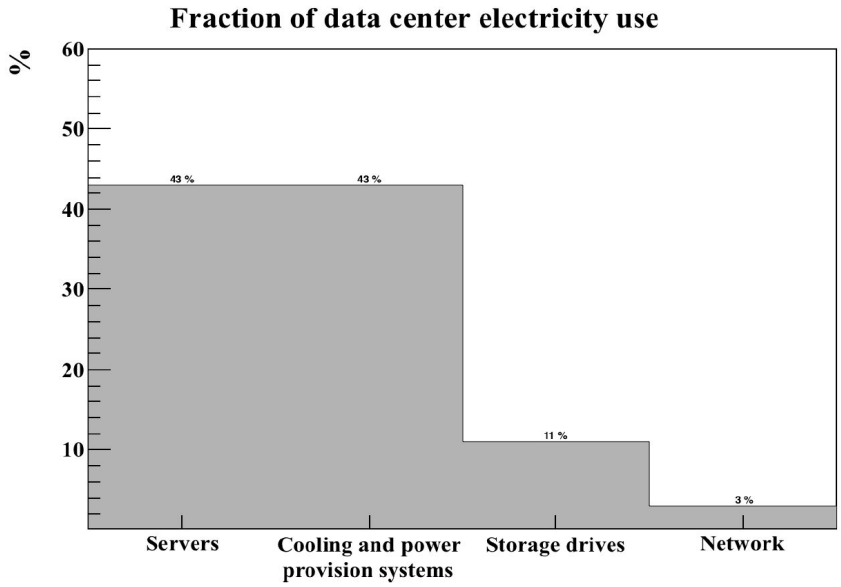


Figure 2: Energy consumption in a data center.

2. Hardware solutions

Physics collaborations employ data centers to analyze the vast amounts of data that come from the experimental apparatuses. In this way, one might say that data centers are indispensable for modern experiments. They are usually equipped with hundreds of processors, allowing users to submit several computing tasks in parallel, minimizing the amount of time required for a single study. Data mass processing and Monte-Carlo file generation also rely on data centers.

In the particular case of the KM3NeT collaboration, two data centers (mainly) play an essential role. These are:

1. the Centre de Calcul (CC) of IN2P3 (Lyon, France) [3], and
2. the INFN CNAF data center (Bologna, Italy) [4].

Although, for the future, several other sites are considered for which the scheme for distributed computing is currently set up; see for instance Ref. [5]. Furthermore, it should be noted that individual institutes are usually equipped with small-scale computing centers. Both CC and CNAF consume significant amounts of power while operating, resulting in the emission

of GHGs and CO_2 as already explained. Fig. 2 shows the main components of a computing center and the fraction of energy that they consume. It comes as no surprise that servers require a large amount of a data center's energy, while it is interesting that power provision systems (UPS, etc.) and cooling units require roughly 43 % of the total energy.

***Figure 3**: The Schneider AP7822 PDU [**6**].*

a. Energy measurement and monitoring

The first step that one should take towards the reduction of CO_2 emissions is to understand and measure where the energy is consumed in a data center. As a rule of thumb: *one cannot reduce what he, or she, cannot measure*. Fortunately, this task can be accomplished with the installation of smart power distribution units (PDU) at the rack level. As an example, we can mention the **Schneider AP7822** PDU model [6] (Fig. 3) that comes with software that helps to measure and monitor energy consumption. In this way, one can identify power hogs, idle equipment and, in general, understand where improvement is possible. For more thorough analyses, several data center infrastructure management (DCIM) software exist that allow one to monitor all the important parameters of a data center. See for instance the package available by **Sunbird** [7]. To set up these packages it might require some extra hardware and software resources, but it is strongly recommended that both CC and CNAF should use it, having the advantages already emphasized.

Servers consume energy even when they are not utilized to do a specific task. Identifying which part of the equipment is idle, or lightly used, can be helpful in mainly two ways. First, one has the option to power down idle servers. Since they consume energy even when not in use, there's no need for them to function. Second, and what is more important, one has the possibility to *consolidate* lightly used servers. Server consolidation is a process where two or more servers are combined together to perform a single job [8]. This recipe

allows an energy efficient use of the computing resources and reduces energy consumption since it reduces the number of physically used servers.

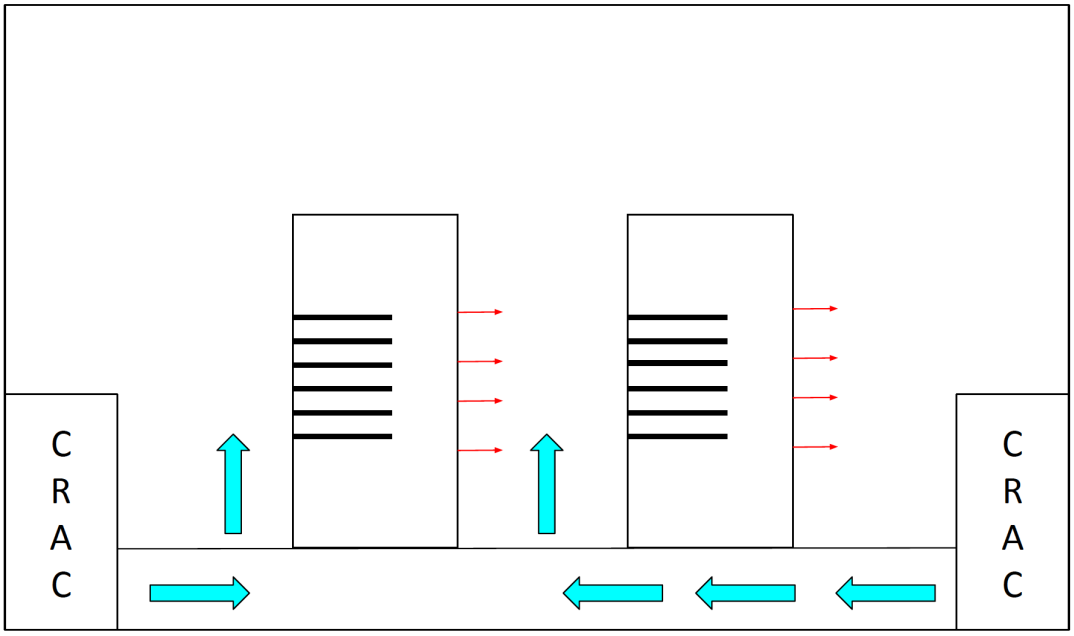


Figure 4: *Airflow management.*

b. Airflow management

Control room air conditioning (CRAC) units consume a significant amount of a data center's energy as it is shown in Fig. 2. Most data center's administrators choose to operate the equipment in a temperature of about 18 ^{o}C. Nonetheless it is a known fact that modern equipment can do well in higher temperatures, e.g. at a temperature level of about 25 ^{o}C. A sizable reduction of energy consumption could be achieved by simply increasing ambient temperatures. Literature suggests that every degree of increase brings about ~ 4 % of savings, Ref. [9].

A standard configuration of a data center's air conditioning and airflow design is shown in Fig. 4. CRAC units are placed at the corners of the room while cold air (blue arrows) is exhausted in front of the racks, passing under an elevated floor. Here it is important that all obstacles are removed, and nothing blocks the cold air flow. Now, typically, fans installed at the back of the racks exhaust the hot air from the equipment into the room (red arrows). Usually, fans operate constantly but, as a matter of fact, there is no need for them to function during periods of low-CPU utilization. Research has shown that switching to variable-speed fans (operating only when needed) one is able to significantly decrease energy usage. A drop of ~ 20 % could be achieved through CPU fan speed reduction alone; see Ref. [10].

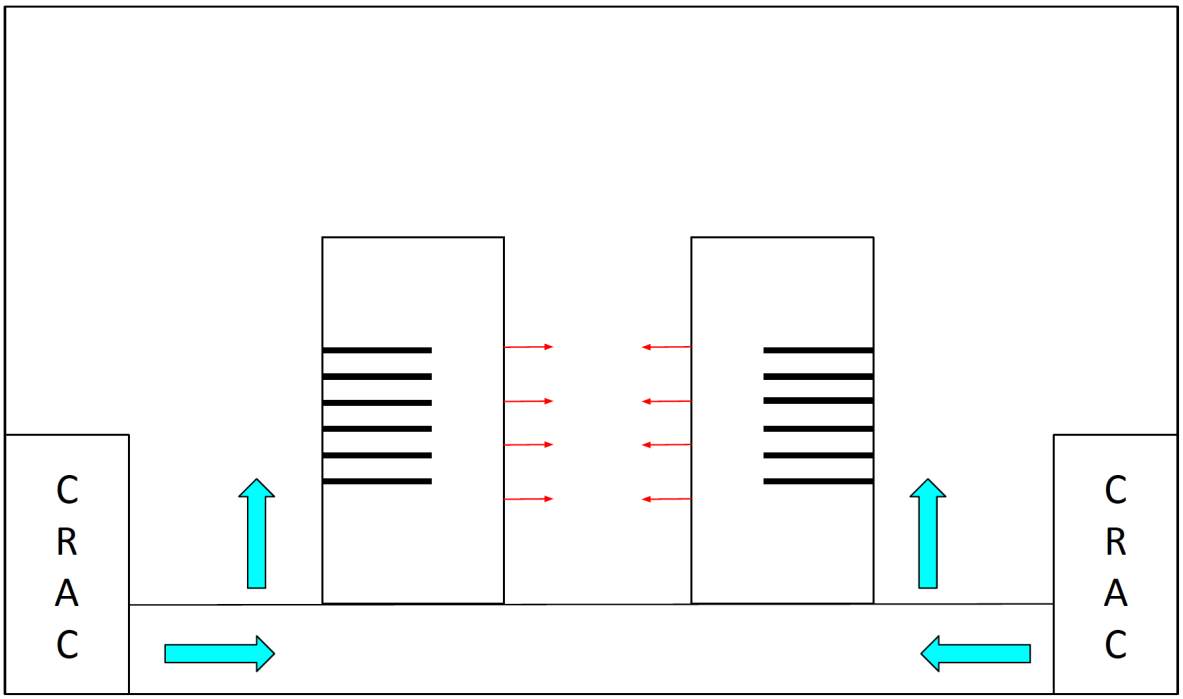


Figure 5: *Hot aisle/cold aisle configuration.*

c. Hot aisle containment

A significant amount of the energy, consumed by the air conditioning units, is required to cool down the hot air exhausted at the back of the racks. This situation is graphically depicted in the middle of Fig. 4. Nonetheless, it is nowadays recognised that this layout is somewhat impractical. For one thing, one has to cool down air (needed for the hardware) that is already cold. This is a waste of energy. A more energy-efficient strategy can be put forward utilizing the so-called *cold aisle/hot aisle configuration*.

This recipe was first used by IBM in 1992. It entails lining up server racks in alternating rows: the fronts of the servers facing each other, creating thus a *cold aisle*, and likewise the backs of the racks face each other producing a *hot aisle*. This layout is shown in Fig. 5. Adhering to this practice, one prevents the cold air from mixing with the hot and it can reduce energy consumption by a factor of 20 to 25 % [11]. Hot exhaust air rises from server racks in the hot aisle and returns along the ceiling back to the CRAC unit intakes. Please notice that this layout requires no funds at all. It is a simple rearrangement of already existing equipment.

Another step towards the improvement of the CRAC system can be achieved exploiting the so-called *hot aisle containment* configuration. In this design the hot aisles created at the back of the server racks are isolated from the rest of the computer room. Simple (and inexpensive) plastic panels can be used for this purpose. Like this, there is a physical barrier between the cold and hot corridors, and cold air is prevented from mixing with the hot in a more drastic manner. Fig. 6 shows a picture of the hot aisle containment configuration adopted by Google corporation. One can clearly see the plastic door separating the hot and

cold regions of the data center. In this way it is more efficient for the cold air to reach the equipment avoiding mixing with the exhaust hot air at the back of the racks.

Now, in the hot aisle containment layout there are three possibilities for dealing with the heat waste contained in the hot corridor:

1. one can direct (with exhaustion pipes) the hot air in the CRAC units intake, where it gets cooled down again and it is recirculated in the front of the servers,

Figure 6: *Google's hot aisle containment configuration.*

2. it can be fanned out of the building and,
3. it can be exploited to perform some task. For instance, the Equinix data center in Paris uses the heat waste to warm a nearby pool [12].

We would like to emphasize (once more) that the air conditioning design that we propose for the KM3NeT data centers (both for the Tier2 and on-shore computing centers) requires no extra funds. The hot aisle configuration can be implemented fairly easily, since it is a simple rearrangement of the existing hardware with the addition of a few blanking panels. The fact that a leading-edge company, like Google, has chosen this layout for its data centers, only serves to validate our point.

d. Cooling coils

The choice of the CRAC units technology, employed in a data center, is rather vital. For one thing, it drives the costs of putting the CRAC system together and a smart decision can reduce drastically energy consumption. During the several stages of this work package, many ideas were considered, weighing in the advantages and disadvantages of each technology separately. At the end of the day, we came to the conclusion that the simple solution of

cooling coils is the most robust. Fig. 7 shows a few pictures of such cooling coils available by Air Systems [13].

Cooling coils are devices cheap to purchase and easy to operate. The primary principle of their operation is based on the fact that cold liquid is circulated through the copper pipes while fans blow the cold air in the computer room. It relies on liquid cooling technology which is an efficient way to achieve low temperatures and it is known to be superior, compared to dry air methods. A cooling tower is typically installed outside of the computer room, where the liquid is cooled down to the desirable temperatures and it is recirculated back in the coils.

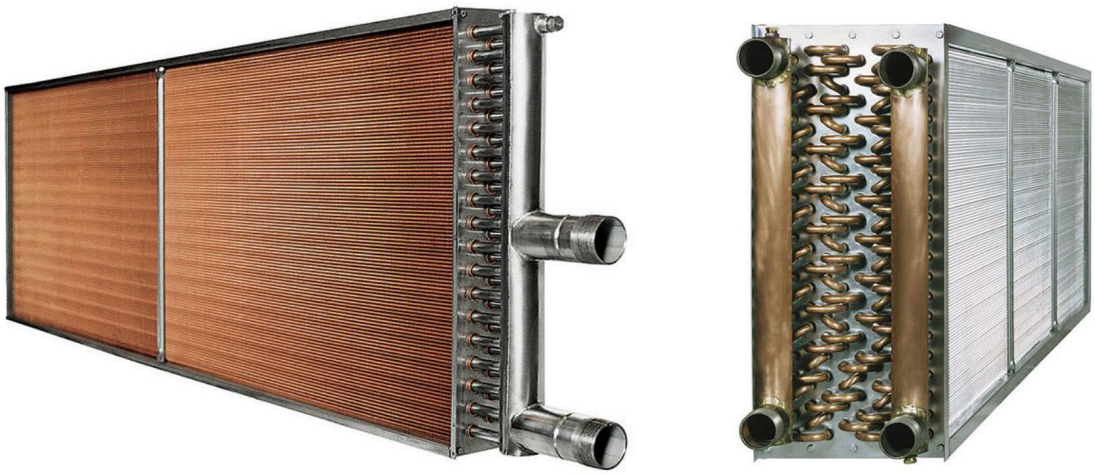


Figure 7: *The Air System's cooling coils [**13**].*

The CRAC scheme that we propose can greatly benefit from cold climates. There are several ways it can be done. The most common is to immerse the cooling towers in a natural reservoir of cold water, for instance, in the cold waters of a nearby river. Several green data centers have adopted this recipe, eliminating in the process the electric energy that is required for the cooling towers to function. As an example, we may mention the computing center, built by Schneider Electric and GRNET, that exploits for this purpose the cold waters of Louros river [14]. Fig. 8 shows the cooling tower of this data center immersed in the waters of the river.

3. Software solutions

In the previous section, we examined several solutions that can reduce the energy usage of a computing center and thus minimize the associated CO_2 emissions. All these ideas were focused on things that can be done at the hardware level. In this section, we shall propose a new handful of ideas that can be implemented at the software level and towards the same objective. Notice that the proposals we put forward in this section are very important since they can be carried through without the need to purchase any additional equipment. They require no extra funds besides the human resources that are necessary to enact them.

Figure 8: *Cooling tower immersed in the waters of Louros river.*

a. Storage drives

Storage drives require a significant amount of a computing center's energy as can be seen in Fig. 2. They consume a portion of ~ 11 % of the total energy budget. This is indeed rather pronounced. Also, it should be noted that they require electric power even when the server is idle. Nonetheless, modern hard drives come with a feature to deactivate after a certain amount of time and when the device is not in use. This is an option that is not enabled by default. Setting the time interval to, say, something like ten minutes one can remarkably reduce energy consumption in times of low-CPU usage.

b. Server virtualization

On the other hand, **server virtualization** is a powerful tool towards green or sustainable computing [15,16]. Traditionally, in a computing center, a single machine is allocated to perform a single task, e.g. a server is employed to host the website or store the various databases, i.e. one workload for one server. This recipe is wasting unused processors, memory capacity and other hardware resources such as network bandwidth. Server virtualization allows one to depart from this simplistic picture and allows for a single server to perform several workloads.

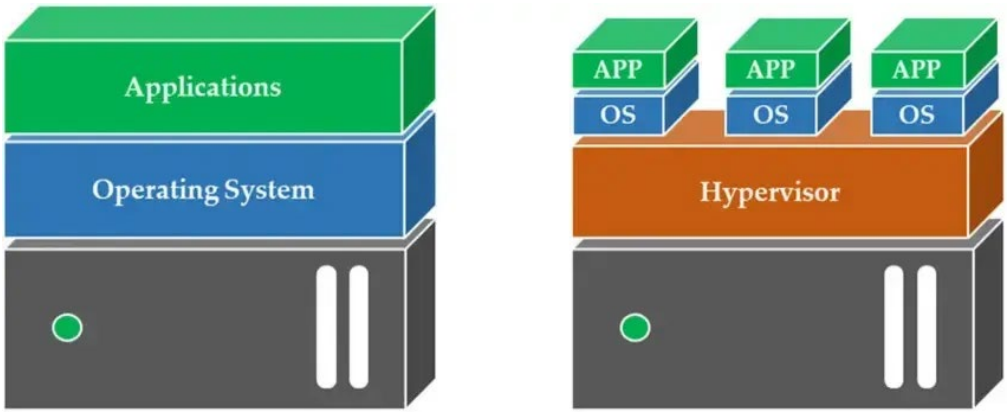


Figure 9: Server virtualization.

Server virtualization is a process that creates and abstracts multiple unique, and isolated, virtual instances on a single physical server. To state it differently, several virtual servers can run in parallel while hosted at the same machine. The various virtual instances are managed by a software called **hypervisor**. Each virtual server can run its own operating system (OS) independently. Now, each independent virtual server can run a single application (APP). The whole process is graphically depicted in Fig. 9. The result should be clear: within this scheme a single server can execute several workloads in parallel.

It is perhaps difficult to overemphasize the importance of server virtualization in green computing and in computing in general. For one thing, one can make for a better use of the resources available in a data center, improving drastically operational efficiency. Moreover, it eliminates the need to acquire new hardware, reducing purchase funds and function costs. This includes not only the cost of buying new servers but also the expenses associated with maintenance, cooling, and physical space requirements. Additionally, it lowers energy consumption in a computing center since the same tasks can be done with less hardware, with no need to acquire new equipment. It also diminishes e-waste generation.

Within the KM3NeT Collaboration both CNAF and CC Lyon computing farms adhere to the logic: one workload for each node. Adding the concept of server virtualization might require some extra work from the information and technology (IT) staff, but it will have all those advantages highlighted in this section.

Last, we should mention that load balancing is nowadays a standard tool in server setup and it is employed also for our KM3NeT services. Say, for those services hosted at the ECAP computing server. For instance, one can design jobs such that they are distributed over several machines. Actually, it should be noted that multithreading has been implemented in the official KM3NeT software to that end, and the new workflow implementations are also designed for that. So the jobs are designed to be flexible in execution and exploit the available computing resources of a computing node. Nonetheless, adding server virtualization will add

a new layer to these already existing functionalities, allowing several servers to run on a single machine.

c. *Programming language*

The choice of the programming language adopted by a physics collaboration is rather fundamental. Nowadays, it is a fact that most particle physics experiments use either *C++* or *python*. This excludes the software that is needed for the Data Acquisition (DAQ) system or specific firmware required for the electronics. C++ offers an object-oriented framework with several container classes that can be used for data storage. On the other hand, python provides a simple, interpreted language that can be utilized for high-level analyses. One should not fail to overemphasize the fact that many advanced machine learning (ML) algorithms are written in python.

Nevertheless, it is a known fact that compiled languages like C, C++ and Java require far less CPU than interpreted languages. This list includes bash script, Perl, Ruby and python among others. In general, it has been estimated that python, despite its popularity and ease of use, consumes up to (approximately) fifty times more energy than C++ for equivalent tasks. Ruby and Perl, Lua, and TypeScript also fall into the category of high-energy consumers [17].

The authors of Ref. [18] investigated several aspects of software development towards the sustainability requirement. In the aforementioned collection of papers various key elements of each programming language were treated. In particular, the authors of the paper entitled "*Save the Earth, Program in C++!*" single out C++ as the most appropriate language in connection to green computing. Several other options were considered and (again) it was found that compiled languages outperform interpreted ones. C++ nonetheless is the language that was found to require less amounts of energy to perform equivalent tasks.

In the particular case of the KM3NeT collaboration, two software packages are (mainly) used in data analyses. These are the AANET and JPP. The core code of both this software is written in object-oriented style, and it is based in C++ and ROOT alone. In this respect, one might say that the analysis flow of KM3NeT is such that it provides an energy-efficient framework and reduces (at its best) the energy required to perform a single task. In general, we should repeat that for the standard low-level analyses, where large volumes of data are involved, the use of C++ is compulsory. It is the language that takes the least execution time and consumes the least amounts of energy. For lighter high-level studies, involving small volumes of data, interpreted languages like python can be used. Additionally, for several high-level analyses, where specific ML algorithms are required (that exist only in python) the use of python is imperative. Note also that there is a new trend in programming (also currently under investigation in KM3NeT) that uses *Julia*; a language that is a bridge between C++ and python using the best features from both languages.

d. *Compiler options*

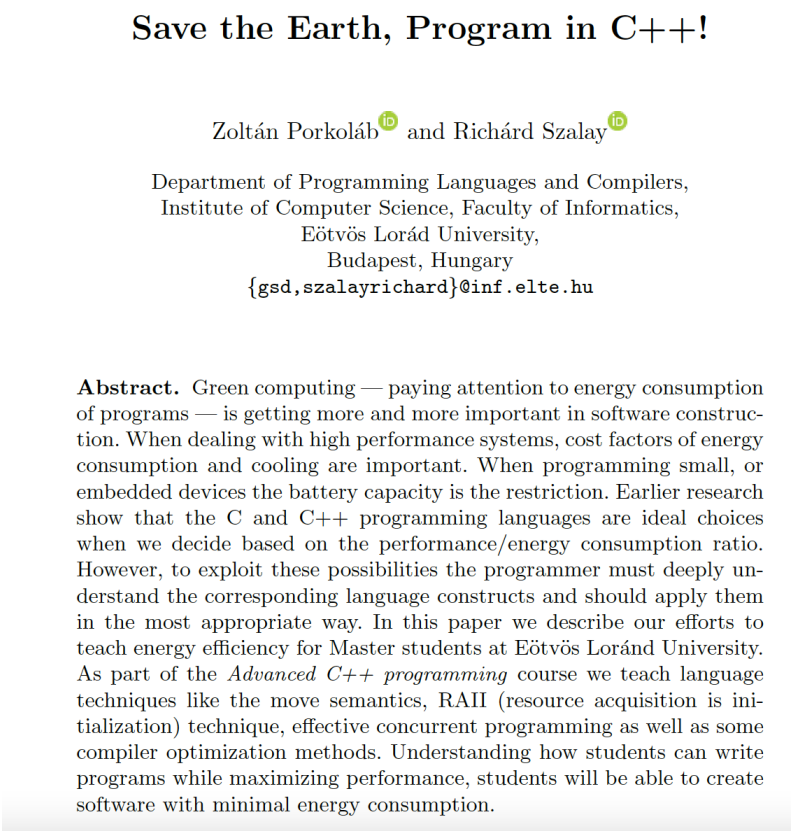
In compiled languages the arguments passed to the compiler do play an essential role. We have already singled out C++ as the best language with respect to green computing, and so will focus our attention to C++ alone in this subsection, as this is also the language mainly employed in KM3NeT. With C++, several arguments can be inserted at the compiling process. We have considered many options but at the end we find that the argument more relevant for this study is the **-Werror**. The -Werror flag considers all warnings in the software as errors, causing the compilation stage to fail. State it differently, all warnings are counted as errors. This is a general advice for better programming since it excludes parts of code that are not used and (in a similar way) eliminates unused variables. This is a notable reduction of the memory utilized by any piece of code. Additionally, it removes allocated pointers that have no function in the software, and frees up their associated memory.

As of now, the KM3NeT software packages do not adhere to this logic and the -Werror flag is not activated. Adding this option to the compiler might require some extra work (in particular one will have to remove all current warnings), but utterly one gains all the advantages highlighted in this subsection. Similar arguments hold for other compiled programming languages as well.

We should also mention that the KM3NeT collaboration has general guidelines for software quality. These include implementation of tests, containerization of the software and benchmarking, employing measurements for executing time. This is (of course) a wider scheme than using a compiler flag but the use of this simple argument in the compilation stage will only help to strengthen the quality of our software.

e. *Data format*

The raw data collected by the KM3NeT detectors arrive offshore in binary format. Then they pass from the event builder routine where they are transformed to ROOT TTrees [16]. Only elementary information is stored there. This is the triggered photomultiplier tube ID number, the hit time and the time over threshold (TOT). Afterwards, calibration constants are applied and the sophisticated reconstruction algorithms are run. From the fully reconstructed data, two separate reduced files are created: these are **Data Summary Tree** (DST) and the **h5** files. The former are based on ROOT classes, while the later are used by python scripts.



Save the Earth, Program in C++!

Zoltán Porkoláb and Richárd Szalay

Department of Programming Languages and Compilers,
Institute of Computer Science, Faculty of Informatics,
Eötvös Lorád University,
Budapest, Hungary
{gsd,szalayrichard}@inf.elte.hu

Abstract. Green computing — paying attention to energy consumption of programs — is getting more and more important in software construction. When dealing with high performance systems, cost factors of energy consumption and cooling are important. When programming small, or embedded devices the battery capacity is the restriction. Earlier research show that the C and C++ programming languages are ideal choices when we decide based on the performance/energy consumption ratio. However, to exploit these possibilities the programmer must deeply understand the corresponding language constructs and should apply them in the most appropriate way. In this paper we describe our efforts to teach energy efficiency for Master students at Eötvös Loránd University. As part of the *Advanced C++ programming* course we teach language techniques like the move semantics, RAII (resource acquisition is initialization) technique, effective concurrent programming as well as some compiler optimization methods. Understanding how students can write programs while maximizing performance, students will be able to create software with minimal energy consumption.

Figure 10: **Green computing with C++.**

DST and h5 files are much lighter than the original reconstructed ones and they are more appropriate for high-level analyses. Since they have, in general, smaller sizes a straightforward analysis is expected to run faster, minimizing in this process the energy required to perform a single study. Notice though a redundancy: the DST and h5 usually hold the same information for the events that trigger the KM3NeT detector(s). That is, the same information is stored twice in two separate files. This is unfortunate, but necessary since our current analysis chain requires the use of h5 files for several analyses that use specific ML algorithms that are only available in python.

To minimize the energy required for a specific workload (and reduce as well the associated CO_2 emissions), the first thing that needs to be done is to reduce the size of the analysis files. In this sense, we propose that only the information needed for the analyses is stored in DST and h5. Avoid introducing redundant variables in the data containers. Additionally, in several cases the computational accuracy of several variables can be limited. For instance, say, that a variable is stored in double format in a DST tree but this precision is not needed for one study. One can lower the accuracy to a float, in reduced versions of the data format, without affecting the results. This practice, with no doubt, will decrease the quota of input files and reduce execution time.

Third, and what is even more important: include all variables needed for a specific study in the DST and h5 files. Avoid calculating numbers online during an analysis. At first glance, this might seem contradictory since the introduction of more information will certainly increase the size of the files. Nonetheless, experience shows that in many cases, computing additional variables on the fly increases dramatically the execution time, rising in parallel energy consumption. All information should be calculated once, at the time where the data files are produced.

In the particular case of DST files there is another possibility. ROOT offers the possibility to deactivate TTree branches (at the preference of the user) using the **SetBranchStatus** function [19]. Say that a TTree has one hundred branches, but only twenty of them are needed for some study. One can deactivate eighty branches, while looping over the events, eliminating the I/O memory needed to read those suppressed branches. A few figures might be helpful. We analyzed a DST file with one hundred and twenty branches. For some dummy analysis this took two minutes and forty-five seconds. Activating only twenty branches, it took (for the same analysis) forty seconds alone: an almost 75 % reduction in execution time. A similar decrease in energy consumption is to be expected.

4. Data processing

We must repeat that the main function of a computing center in a physics experiment is to process the data coming from the detector(s). The volume of the information is (usually) so large and it cannot be analyzed without the help of a computing farm. On the other hand, one would prefer to process these datasets only once for each specific analysis task. Reprocessing the same data many times, due to user mistakes or other deficiencies, will require substantial amounts of energy and will result in unnecessary emissions of CO_2. Running the heavy reconstruction algorithms only one time (for each study) minimizes the amount of GHGs produced by this process. In any case, we must add that this recipe makes for a better use of our computing resources.

The data come from the detectors in binary form. Subsequently they are transformed (by the event builder) into ROOT format. This is the main data container type of KM3NeT. Then calibration constants are applied. It is important, in this step, to make sure that the correct values are inserted and in the proper way. For this purpose, we propose to look at several plots that can reveal whether the calibration stage is correctly done. This can include the energy resolution, the pointing resolution, etc. After this stage, the sophisticated reconstructed algorithms are run and the quality scripts are applied to the data. Here runs are categorized according to their quality and it is decided which ones are necessary for reprocessing according to their usability for analysis.

Here we propose a novelty. Instead of running the reconstruction stage on the full dataset, we suggest the idea of building only a fraction of the sample. *Say, 5 % of the whole set*. In this way, the analysis chain can be continued, using this limited sample. One can then go on and

perform a top to bottom study. For this purpose, Monte Carlo (MC) files can be produced only for the data that belong to this smaller list. Control parameters can be checked, and all the necessary variables (needed for the analysis) can be plotted and be overlaid with the corresponding MC distributions. One can then check for discrepancies that could tell a possible mistake. Finally, a toy result can be provided. Performing a full-scale analysis on this 5 %, could reveal whether an error has been propagated, avoiding the need to reprocess the whole dataset many times again, wasting significant amounts of energy.

It is important in this step to compare distributions between different versions of the software. A large discrepancy between versions is always the smoking gun for a bug being introduced in the code. Once all checks are met, one can then proceed and process the whole dataset. Note, also that the procedure that we have outlined can be applied to the MC files production *mutatis mutandis*. A list of further cross-checks that are envisioned by the Data Management group can be found in Ref. [5].

Last, it is also necessary to learn from our mistakes. Every time that an error is found, is it important to keep a blacklist and document the errors that have occurred. One can then add a detailed procedure (in the official routines) that can be followed to prevent this error from being propagated again.

5. User recommendations

In this section we will cover the actions that individual users can take towards the sustainability requirement. In particular, we will offer three proposals concerning what the KM3NeT collaborators can do towards the reduction of energy consumption and carbon dioxide emissions.

a. Online meetings

In a physics experiment it is important for researchers to meet and discuss the progress of the project. In large-scale, this is done in the so-called collaboration meetings (CM) where the whole experiment is gathered at a designated place to evaluate the advance of the work, taking into account guidelines and the several deadlines for each deliverable. For short-run, day-to-day activities, the various working groups organize weekly or bi-weekly meetings. Of course, it is only natural that these frequent meetings will take place in an online room. Now, most scientists prefer to turn on their cameras in these encounters. While this might seem (at first glance) rather innocent, we find that it is such a waste of energy. For one thing, the use of a camera does not add to the discussion or the physics arguments. We feel that powering down all private cameras on these occasions will only save a rather substantial amount of energy that can be otherwise used.

b. Dark mode

The essence of data analysis is software development. Experimental physicists use complicated pieces of code to conduct their studies. Now, it is well-known that code editing and development proceeds with the use of an application editor. Several commercial editors are available, and each researcher has their own predilections. Most of them employ a white background with black letters, but there are many editors that offer the possibility to code in a black background with white letters. This **dark mode** requires less brightness by the screen, and it naturally reduces the energy consumption of a desktop or a laptop computer. Adhering to this recipe, users can contribute collectively in reducing GHG emissions in KM3NeT. Note that this idea is not new. Google offers the possibility of a dark theme along the lines that we have explained. It has been argued that when using auto-brightness the energy saving for dark mode is between 3 and 9 %. If screen brightness is set to 100 % the savings can be as high as 47 % [20].

c. Debugging and analysis

Electrical power is also consumed by individual users performing their own analysis tasks. This is a non-negligible amount of energy for every physics experiment. Now, ideally one would like to minimize this number. To this end, we propose that all software (employed for analysis studies) should be first thoroughly checked and debugged. For this need, a list of control variables should be decided, and an analysis should proceed only when a specific number of criteria is met. These checkpoints should depend on the task at hand. Second, and what is even more critical, we propose that all studies are first conducted in a smaller portion of the whole sample, e.g. 5 % of the full dataset. Like this, one reduces drastically the amount of energy required when exercising the analysis scripts, since we now run on a limited part of the official runlist. Once all cuts and code are “frozen” one can then proceed to exercise the full sample. Note that this procedure resembles closely blinding techniques using a so-called *burn-up sample*. Although with a notable difference: one does not need to reject this 5 % after the analysis is concluded.

6. Renewable energy sources

In this report we have made several proposals that could be easily adopted by the KM3NeT collaboration towards the reduction of the GHG emissions related to its computing resources. We elaborated on both hardware and software solutions. We have also presented a data processing model that aims to reduce the energy consumption from multiple reruns of the high-level reconstruction algorithms. User recommendations were also considered. Another possibility that should be examined would be to power up data centers using so-called renewable energy sources. State it differently, buy (for this purpose) green energy from vendors that sell it. Fig. 11 shows the grammars of CO_2 emitted per kWh for various energy sources. One sees that coal, oil and natural gas have rather strong contributions. On the other

nuclear, solar, wind and hydropower have remarkably low rates of CO_2 intensities. Taking the decision, or the choice, to use these environmentally friendly energy sources to operate a data center would have a dramatic reduction of GHG emissions. This is an option that should be seriously taken into account by the KM3NeT Collaboration.

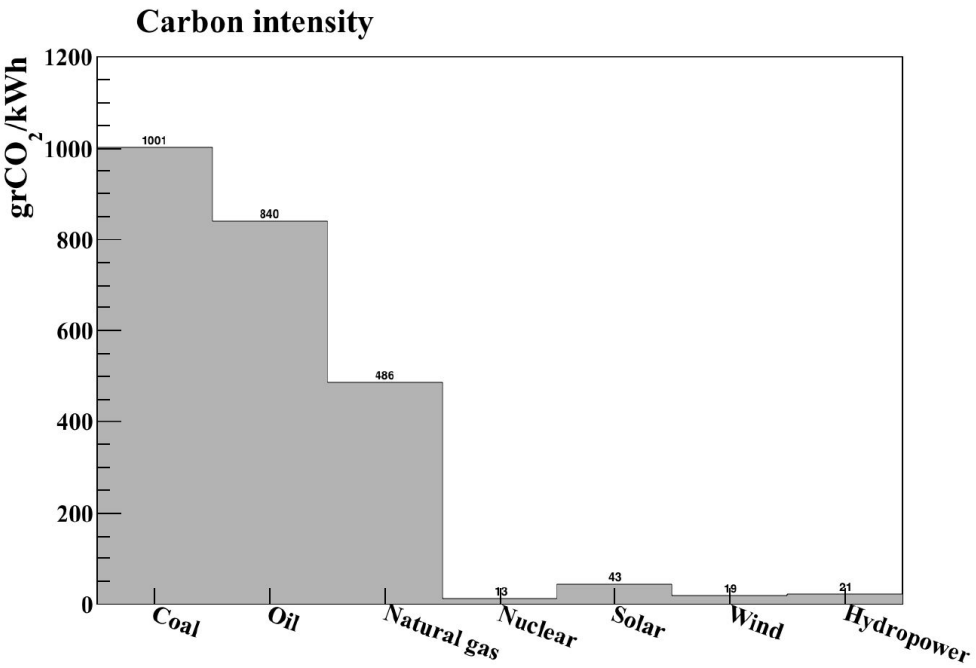


Figure 11: **Carbon dioxide emission per energy source.**

7. Summary

In this last section, we try to summarize the key recommendations that we have presented throughout this technical report:

a. Hardware solutions

1. The first step towards the reduction of the energy consumption of a data center is to monitor where the energy is consumed. As a rule of thumb: one cannot reduce what he, or she, cannot measure. Smart PDUs offer this possibility and detailed DCIM software packages exist that can be employed to monitor all the relevant parameters of a data center.
2. Server consolidation, of lightly used servers, is a recipe that makes for a better use of our hardware resources and reduces energy consumption from underutilized hardware.

3. Most data center's administrators choose to operate the equipment in a temperature of about 18 ^{o}C. Nonetheless it is a known fact that modern equipment can do well in higher temperatures, e.g.. at a temperature level of about 25 ^{o}C. A sizable reduction of energy consumption could be achieved by simply increasing ambient temperatures.
4. The airflow management in a data center plays a rather important role. Here administrators must make sure that all obstacles are removed, and nothing blocks the cold air flow.
5. The use of variable-speed fans (operating only when needed) significantly decreases energy usage. A drop of ~ 20 % could be achieved through CPU fan speed reduction alone.
6. The cold aisle/hot aisle configuration of the data center's air conditioning layout reduces energy usage by a factor of 20 to 25 % in an efficient and cost-effective way.
7. Hot aisle containment further improves on the air conditioning design, preventing cold air to mix with hot air through a physical barrier. A similar decrease in energy consumption should be expected.
8. Cooling coils offer a simple choice for data center's CRAC units. They are cheap to purchase and easy to operate. Note also, that they can greatly benefit from cold climates. For instance, the cooling towers can be immersed in the cold waters or a nearby river, eliminating the costs for the towers to operate and cool down the liquid circulated through the coils.

b. Software solutions

1. Modern hard drives come with a feature to deactivate after a certain amount of time and when the device is not in use. This is an option that is not enabled by default. Setting the time interval to, say, something like ten minutes one can remarkably reduce energy consumption in times of low-CPU usage.
2. Server virtualization is a powerful tool in green computing. For one thing, one can make for a better use of the resources available in a data center, improving drastically operational efficiency. Moreover, it eliminates the need to acquire new hardware, reducing purchase funds and function costs. This includes not only the cost of buying new servers but also the expenses associated with maintenance, cooling, and physical space requirements. Additionally, it lowers energy consumption in a computing center since the same tasks can be done with less hardware, with no need to acquire new equipment. It also diminishes e-waste generation.
3. Interpreted languages consume up to fifty times more energy than compiled ones to perform equivalent tasks. Here it is recommended that for

computationally expensive studies, the KM3NeT collaboration should employ a compiled language. In particular, C++ has been found to be the most effective one.

4. Setting the compiler options to count warnings as errors makes for a better computing practice in general. More specifically, it excludes parts of code that are not used and (in a similar way) eliminates unused variables. This is a notable reduction of the memory utilized by any piece of code.
5. Whenever possible, limit the computational accuracy of a variable. This practice will lower the size of the data files, and will decrease the energy needed for a high-level analysis.
6. Avoid calculating numbers online during an analysis. Experience shows that in several cases, computing additional variables on the fly increases dramatically the execution time, rising in parallel energy consumption. All information should be calculated once, at the time where the data files are produced.
7. When analysing ROOT ntuples consider to deactivate TBranches that are not needed for this specific analysis. For several studies, a decrease of ~ 75 % in execution time can be achieved by this simple process.

c. Data processing

1. After calibration constants are applied it is important to make sure that the correct values are inserted and in the proper way. For this purpose, we propose to look at several plots that can reveal whether the calibration stage is correctly done. This can include the energy resolution, the pointing resolution, etc.

2. Instead of running the reconstruction stage on the full dataset, we suggest the idea of building only a fraction of the sample. *Say, 5 % of the whole set*. In this way, the analysis chain can be continued, using this limited sample. One can then go on and perform a top to bottom study. Control parameters can be checked, and all the necessary variables (needed for the analysis) can be plotted and be overlaid with the corresponding MC distributions.

3. Compare distributions between different versions of the software and the MC. A large discrepancy between versions is always the smoking gun for a bug being introduced in the code.

4. It is important to keep a blacklist and document the errors that have occurred. One can then add a detailed procedure (in the official routines) that can be followed to prevent this error from being propagated again.

d. User recommendations

1. Switch off cameras in online meetings. The use of a camera does not add to the discussion or the physics arguments. We feel that powering down all private

cameras on these occasions will only save a rather substantial amount of energy that can be otherwise used.

2. Employ dark mode in the software editors. It has been found that when using auto-brightness the energy saving for dark mode is between 3 and 9 %. If the screen brightness is set to 100 % the savings can be as high as 47 %.
3. For all private high-level analyses, run first on a limited list of the whole sample. Once all cuts and procedures are “frozen” one can then run on the whole runlist.

e. Renewable sources of energy

1. Consider the possibility to power up data centers using so-called renewable energy sources. State it differently, buy (for this purpose) green energy from vendors that sell it. Taking the decision, or the choice, to use these environmentally friendly energy sources to operate a data center would have a dramatic reduction of GHG emissions

In this technical report, we have presented a number of steps that can be taken towards sustainable or green computing. These recipes reduce energy consumption and minimize carbon dioxide emissions. Obviously, the next step will be to contact the administrators of the data centers and see which recommendations are relevant for their specific data center and valuable to implement. We feel that the proposals made in this document will help the KM3NeT computing centers to reduce carbon dioxide emissions and become more environmentally friendly.

X. REFERENCES

[1] Wikipedia. (n.d.). "Climate change". Accessed on the 16/12/2024 at: https://en.wikipedia.org/wiki/Climate_change

[2] AKCP. (2023). "Data Center Energy Use Trends and Efficiency Strategies". Accessed on the 16/12/2024 at: https://www.akcp.com/blog/the-real-amount-of-energy-a-data-center-use/

[3] CCIN2P3. (2021). "IN2P3 Computing Centre". Accessed on the 16/12/2024 at: https://cc.in2p3.fr/en/

[4] INFN. (n.d.). "CNAF". Accessed on the 16/12/2024 at: https://www.cnaf.infn.it/en/

[5] INFRADEV2 Outputs. "Report on the review of the KM3NeT Data Management Plan by external experts". Available at:

https://www.km3net.org/wp-content/uploads/2024/04/KM3NeT-INFRADEV2-WP4-D4.1-v.approved.pdf

[6] Schneider Electric. (2022). "PDU unit for rack, calibrated". Accessed on the 16/12/2024 at: https://www.se.com/gr/el/product/AP7822/

[7] Sunbird. (n.d.). "Plan, provision, model, and monitor all infrastructure across all sites". Accessed on the 16/12/2024 at: https://www.sunbirddcim.com/

[8] Science Direct. (2015). "Server Consolidation". Accessed on the 16/12/2024 at: https://www.sciencedirect.com/topics/computer-science/server-consolidation

[9] Sunbird DCIM. "What temperature should your data center be ?". Accessed on the 27/12/2024 at: https://www.sunbirddcim.com/blog/what-temperature-should-your-data-center-be

[10] Tech target. "6 ways to increase energy efficiency in data centers". Accessed on the 27/12/2024 at:

https://www.techtarget.com/searchdatacenter/tip/Four-ways-to-reduce-data-center-power-consumption

[11] Energy Star. (n.d.). "Move to a Hot Aisle/Cold Aisle Layout". Accessed on the 16/12/2024 at: https://www.energystar.gov/products/data_center_equipment/16-more-ways-cut-energy-waste-data-center/move-hot-aislecold-aisle-layout

[12] Sydney Adams, H. (2024). "Excess Data Centre Heat Used to Warm Olympic Swimming Pools", *DataCentre Magazine*. Accessed on the 16/12/2024 at: https://datacentremagazine.com/data-centres/excess-data-centre-heat-used-to-warm-olympic-swimming-pools

[13] Air Systems. (2024). "Heating and Cooling Coils". Accessed on the 16/12/2024 at: www.airsystems-usa.com/products/heating-cooling-coils/

[14] Schneider Electric & GRNET. (2016). "Green Data Center in Louros river". Accessed on the 16/12/2024 at: https://www.youtube.com/watch?v=2axTsWvPd0A

[15] ISmile Technologies. (2023). "Virtualization 101: How Virtualization Technology Can Make IT Services More Sustainable". Accessed on the 16/12/2024 at: https://ismiletechnologies.com/technology/virtualization-101-how-virtualization-technology-can-make-it-services-more-sustainable/#

[16] Gillis, A. S. and Bigelow S. J. (n.d.). "What is server virtualization? The ultimate guide", *TechTarget Network*. Accessed on the 16/12/2024 at: https://www.techtarget.com/searchitoperations/definition/What-is-server-virtualization-The-ultimate-guide

[17] Pottigari, A. (2024). "Energy Efficiency of Coding Languages & Exploring the Impact of Language Choice", Medium. Accessed on the 16/12/2024: https://medium.com/@akshithpottigari/energy-efficiency-of-coding-languages-exploring-the-impact-of-language-choice-c5acc1d33012

[18] Koopman, P.; Lubbers, M. and Fernandes, J.P. (2022). *SusTrainable: Promoting Sustainability as a Fundamental Driver in Software Development Training and Education*. Available at: http://arxiv.org/abs/2204.13993

[19] ROOT Reference guide. (n.d.). "TTree Class Reference". Accessed on the 16/12/2024 at: https://root.cern/doc/master/classTTree.html

[20] Valtech. (2023). "Does dark mode win on sustainability and accessibility?". Accessed on the 16/12/2024 at: www.valtech.com/blog/does-dark-mode-win-on-sustainability-and-accessibility/