

# DCworms - a tool for simulation of energy efficiency in distributed computing infrastructures

K. Kurowski<sup>a</sup>, A. Oleksiak<sup>a</sup>, W. Piatek<sup>a</sup>, T. Piontek<sup>a</sup>, A. Przybyszewski<sup>a</sup>,  
J. Weglarz<sup>a,b</sup>

<sup>a</sup>*Poznan Supercomputing and Networking Center, Noskowskiego 10, Poznan, Poland*

<sup>b</sup>*Institute of Computing Science, Poznan University of Technology,  
Piotrowo 2, Poznan, Poland*

---

## Abstract

In the recent years, energy-efficiency of computing infrastructures has gained a great attention. For this reason, proper estimation and evaluation of energy that is required to execute data center workloads became an important research problem. In this paper we present a Data Center Workload and Resource Management Simulator (DCworms) which enables modeling and simulation of computing infrastructures to estimate their performance, energy consumption, and energy-efficiency metrics for diverse workloads and management policies. We discuss methods of power usage modeling available in the simulator. To this end, we compare results of simulations to measurements of real servers. To demonstrate DCworms capabilities we evaluate impact of several resource management policies on overall energy-efficiency of specific workloads executed on heterogeneous resources.

*Keywords:* simulations, workload and resource modeling, energy-efficiency, data centers

---

## 1. Introduction

Rising popularity of large-scale computing infrastructures caused quick development of data centers. Nowadays, data centers are responsible for around 2% of the global energy consumption making it equal to the demand of aviation industry [13]. Moreover, in many current data centers the actual IT equipment uses only half of the total energy whereas most of the remaining part is required for cooling and air movement resulting in poor Power Usage Effectiveness (PUE) [29] values. Large energy needs and significant

$CO_2$  emissions caused that issues related to cooling, heat transfer, and IT infrastructure location are more and more carefully studied during planning and operation of data centers.

For these reasons many efforts were undertaken to measure and study energy efficiency of data centers. There are projects focused on data center monitoring and management [9][1] whereas others on energy efficiency of networks [7]. Additionally, vendors offer a wide spectrum of energy efficient solutions for computing and cooling [30][25][27]. However, a variety of solutions and configuration options can be applied planning new or upgrading existing data centers. In order to optimize a design or configuration of data center we need a thorough study using appropriate metrics and tools evaluating how much computation or data processing can be done within given power and energy budget and how it affects temperatures, heat transfers, and airflows within data center. Therefore, there is a need for simulation tools and models that approach the problem from a perspective of end users and take into account all the factors that are critical to understanding and improving the energy efficiency of data centers, in particular, hardware characteristics, applications, management policies, and cooling. These tools should support data center designers and operators by answering questions how specific application types, levels of load, hardware specifications, physical arrangements, cooling technology, etc. impact overall data center energy efficiency. There are various tools that allow simulation of computing infrastructures. On one hand they include advanced packages for modeling heat transfer and energy consumption in data centers [28] or tools concentrating on their financial analysis [6]. On the other hand, there are simulators focusing on computations such as CloudSim [4]. The CoolEmAll project aims to integrate these approaches and enable advanced analysis of data center efficiency taking into account all these aspects [3][26].

One of the results of the CoolEmAll project is the Data Center Workload and Resource Management Simulator (DCworms) which enables modeling and simulation of computing infrastructures to estimate their performance, energy consumption, and energy-efficiency metrics for diverse workloads and management policies. We discuss methods of power usage modeling available in the simulator. To this end, we compare results of simulations to measurements of real servers. To demonstrate DCworms capabilities we evaluate impact of several resource management policies on overall energy-efficiency of specific workloads executed on heterogeneous resources.

The remaining part of this paper is organized as follows. In Section 2

we give a brief overview of the current state of the art concerning modeling and simulation of distributed systems, such as Grids and Clouds, in terms of energy efficiency. Section 3 discusses the main features of DCworms. In particular, it introduces our approach to workload and resource management, presents the concept of energy efficiency modeling and explains how to incorporate a specific application performance model into simulations. Section 4 discusses energy models adopted within the DCworms. In Section 5 we assess the energy models by comparison of simulation results with real measurements. We also present experiments that were performed using DCworms to show various types of resource and scheduling technics allowing decreasing the total energy consumption of the execution of a set of tasks. In Section 6 we explain how to integrate workload and resource simulations with heat transfer simulations within the CoolEmAll project. Final conclusions and directions for future work are given in Section 7.

## 2. Related Work

The growing importance of energy-efficiency in information technologies led to significant interest in energy saving methods for computing systems. Nevertheless, studies of impact of resource management policies on energy-efficiency of IT infrastructures require a large effort and are difficult to perform in real distributed environments. To overcome these issues, extensive research has been conducted in the area of modeling and simulation and variety of tools that address the green computing have emerged. The most popular ones are: GreenCloud [10], CloudSim [4] and DCSG Simulator [5].

GreenCloud is a C++ based simulation environment for studying the energy-efficiency of cloud computing data centers. CloudSim is a simulation tool that allows modeling of cloud computing environments and evaluation of resource provisioning algorithms. Finally, the DCSG Simulator is a data center cost and energy simulator calculating the power and cooling schema of the data center equipment.

The scope of the aforementioned toolkits concerns the data center environments. However, all of them, except DCworms presented in this paper, restricts the simulated architecture in terms of types of modeled resources. In this way, they impose the use of predefined sets of resources and relations between them. GreenCloud defines switches, links and servers that are responsible for task execution and may contain different scheduling strategies. Contrary to what the GreenCloud name may suggest, it does not allow

testing the impact of virtualization-based approaches. CloudSim allows creating a simple resources hierarchy consisting of machines and processors. To simulate a real cloud computing data center, it provides an extra virtualization layer responsible for the virtual machines (VM) provisioning process and managing the VM life cycle. In DCSG Simulator user is able to take into account a variety of mechanical and electrical devices as well as the IT equipment and define for each of them numerous factors, including device capacity and efficiency as well as the data center conditions.

The general idea behind all of the analyzed tools is to enable studies concerning energy efficiency in distributed infrastructures. GreenCloud approach enables simulation of energy usage associated with computing servers and network components. For example, the server power consumption model implemented in GreenCloud depends on the server state as well as its utilization. The CloudSim framework provides basic models to evaluate energy-conscious provisioning policies. Each computing node can be extended with a power model that estimates the current power consumption. Within the DCSG Simulator, performance of each of the data center equipment (facility and IT) is determined by a combination of factors, including workload, local conditions, the manufacturer's specifications and the way in which it is utilized. In DCworms, the plugin idea has been introduced that offers emulating the behavior of computing resources in terms of power consumption. Additionally, it delivers detailed information concerning resource and application characteristics needed to define more sophisticated power draw models.

In order to emulate the behavior of real computing systems, green computing simulator should address also the energy-aware resource management. In this term, GreenCloud offers capturing the effects of both of the Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management schemes. At the links and switches level, it supports downgrading the transmission rate and putting network equipment into a sleep mode. CloudSim comes with a set of predefined and extensible policies that manage the process of VM migrations in order to optimize the power consumption. However, the proposed approach is not sufficient for modeling more sophisticated policies like frequency scaling techniques and managing resource power states. DCSG Simulator is told to implement a set of basic energy-efficient rules that have been developed on the basis of detailed understanding of the data center as a system. The output of this simulation is a set of energy metrics, like PUE, and cost data representing the IT devices. DCworms introduces a dedicated interface that provides methods to obtain the detailed information about

each resource and its components energy consumption and allows changing its current energy state. Availability of these interfaces in scheduling plugin supports implementation of various strategies such as centralized energy management, self-management of computing resources and mixed models.

In terms of application modeling, all tools, except DCSG Simulator, describe the application with a number of computational and communicational requirements. In addition, GreenCloud and DCworms allow introducing the QoS requirements by taking into account the time constraints during the simulation. DCSG Simulator instead of modeling of the single application, enables the definition of workload that leads to a given utilization level. However, only DCworms supports application performance modeling by not only incorporating simple requirements that are taken into account during scheduling, but also by allowing specification of task execution time.

GreenCloud, CloudSim and DCworms are released as Open Source under the GPL. DCSG Simulator is available under an OSL V3.0 open-source license, however, it can be only accessed by the DCSG members.

Summarizing, DCworms stands out from other tools due to the flexibility in terms of data center equipment and structure definition. Moreover, it allows to associate the energy consumption not only with the current power state and resource utilization but also with the particular set of applications running on it. Moreover, it does not limit the user in defining various types of resource management policies. The main strength of CloudSim lies in implementation of the complex scheduling and task execution schemes involving resource virtualization techniques. However, the energy efficiency aspect is limited only to the VM management. The GreenCloud focuses on data center resources with particular attention to the network infrastructure and the most popular energy management approaches. DCSG simulator allows taking into account also non-computing devices, nevertheless it seems to be hardly customizable to specific workloads and management policies.

### **3. DCworms**

The following picture (Figure 1) presents the overall architecture of the simulation tool.

Data Center workload and resource management simulator (DCworms) is a simulation tool based on the GSSIM framework [11] developed by Poznan Supercomputing and Networking Center (PSNC). GSSIM has been proposed to provide an automated tool for experimental studies of various resource



Figure 1: DCworms architecture

management and scheduling strategies in distributed computing systems. DCworms extends its basic functionality and adds some additional features related to the energy efficiency issues in data centers. In this section we will introduce the functionality of the simulator, in terms of modeling and simulation of large scale distributed systems like Grids and Clouds.

### 3.1. Architecture

DCworms is an event-driven simulation tool written in Java. In general, input data for the DCworms consist of workload and resources descriptions. They can be provided by the user, read from real traces or generated using the generator module. In this terms DCworms benefits from the GSSIM workload generator tool that allows creating synthetic workloads ([11]). However, the key elements of the presented architecture are plugins. They allow the researchers to configure and adapt the simulation environment to the peculiarities of their studies, starting from modeling job performance, through energy estimations up to implementation of resource management and scheduling policies. Each plugin can be implemented independently and plugged into a specific experiment. Results of experiments are collected, aggregated, and visualized using the statistics module. Due to a modular and plug-able architecture DCworms can be applied to specific resource management problems and address different users requirements.

### 3.2. Workload modeling

As it was said, experiments performed in DCworms require a description of applications that will be scheduled during the simulation. As a primary definition, DCworms uses files in the Standard Workload Format (SWF) or its extension the Grid Workload Format (GWF) [21]. In addition to the SWF file, some more detailed specification of a job and tasks can be included in an auxiliary XML file. This form of description extends the basic one and provides the scheduler with more detailed information about application profile, task requirements, user preferences and execution time constraints, which are unavailable in SWF/GWF files. To facilitate the process of adapting the traces from real resource management systems, DCworms supports reading those delivered from the most common ones like SLURM [22] and Torque [24]. Since the applications may vary depending on their nature in terms of their requirements and structure, DCworms provides user flexibility in defining the application model. Thus, considered workloads may have various shapes and levels of complexity that range from multiple independent jobs, through large-scale parallel applications, up to whole workflows containing time dependencies and preceding constraints between jobs and tasks. Each job may consist of one or more tasks and these can be seen as groups of processes. Moreover, DCworms is able to handle rigid and moldable jobs, as well as pre-emptive ones. Levels of information about incoming jobs are presented in Figure 2.

To model the application profile in more detail, DCworms follows the DNA approach proposed in [8]. Accordingly, each task can be presented as a sequence of phases, which shows the impact of this task on the resources that run it. Phases are then periods of time where the system is stable (load, network, memory) given a certain threshold. Each phase is linked to values of the system that represent a resource consumption profile. Such a stage could be for example described as follows: 60% CPU, 30% network, 10% memory. This form of representation allows users to define a wide range of workloads: HPC (long jobs, computational-intensive, hard to migrate) or virtualization (short requests) that are also typical for data center environments.

### 3.3. Resource modeling

The main goal of DCworms is to enable researchers evaluation of various resource management policies in diverse computing environments. To this end, it supports flexible definition of simulated resources both on physical (computing resources) as well as on logical (scheduling entities) level. This

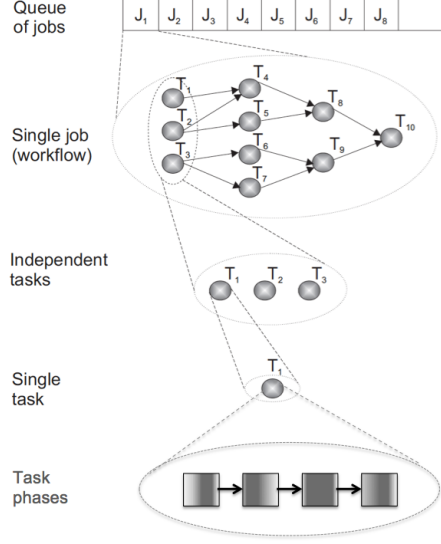


Figure 2: Levels of information about jobs

flexible approach allows modeling of various computing entities consisting of compute nodes, processors and cores. In addition, detailed location of the given resources can be provided in order to group them and arrange into physical structures such as racks and containers. Each of the components may be described by different parameters specifying available memory, storage capabilities, processor speed etc. In this way, it is possible to describe power distribution system and cooling devices. Due to an extensible description, users are able to define a number of experiment-specific and visionary characteristics. Moreover, with every component, dedicated profiles can be associated that determines, among others, power, thermal and air throughput properties. The energy estimation plugin can be bundled with each resource. This allows defining various power models that can be then followed by different computing system components. Details concerning the approach to energy-efficiency modeling in DCworms can be found in the next sections.

Scheduling entities allow providing data related to the brokering or queuing system characteristics. Thus, information about available queues, resources associated with them and their parameters like priority, availability of advance reservation mechanism etc. can be defined. Moreover, allocation



policy and task scheduling strategy for each scheduling entity can be introduced in form of the reference to an appropriate plugin. DCworms allows building a hierarchy of schedulers corresponding to the hierarchy of resource components over which the task may be distributed.

In this way, the DCworms supports simulation of a wide scope of physical and logical architectural patterns that may span from a single computing resource up to whole data centers (even geographically distributed). In particular, it supports simulating complex distributed architectures containing models of the whole data centers, containers, racks, nodes, etc. In addition, new resources and distributed computing entities can easily be added to the DCworms environment in order to enhance the functionality of the tool and address more sophisticated requirements. Granularity of such topologies may also differ from coarse-grained to very fine-grained modeling single cores, memory hierarchies and other hardware details.

#### *3.4. Energy management concept in DCworms*

The DCworms allows researchers to take into account energy efficiency and thermal issues in distributed computing experiments. That can be achieved by the means of appropriate models and profiles. In general, the main goal of the models is to emulate the behavior of the real computing resources, while profiles support models by providing data essential for the energy usage calculations. Introducing particular models into the simulation environment is possible through choosing or implementation of dedicated energy plugins that contain methods to calculate power usage of resources, their temperature and system air throughput values. Presence of detailed resource usage information, current resource energy and thermal state description and a functional energy management interface enables an implementation of energy-aware scheduling algorithms. Resource energy consumption and thermal metrics become in this context an additional criterion in the resource management process. Scheduling plugins are provided with dedicated interfaces, which allow them to collect detailed information about computing resource components and to affect their behavior. The following subsection presents the general idea behind the power management concept in DCworms. Detailed description of the approach to thermal and air throughput simulations can be found in [15].

#### 3.4.1. Power management

The motivation behind introducing a power management concept in DCworms is providing researchers with the means to define the energy efficiency of resources, dependency of energy consumption on resource load and specific applications, and to manage power modes of resources. Proposed solution extends the power management concept presented in GSSIM [12] by offering a much more granular approach with the possibility of plugging energy consumption models and power profiles into each resource level.

**Power profile.** In general, power profiles allow specifying the power usage of resources. Depending on the accuracy of the model, users may provide additional information about power states which are supported by the resources, amounts of energy consumed in these states, and other information essential to calculate the total energy consumed by the resource during runtime. In such a way each component of IT infrastructure may be described, including computing resources, system components and data center facilities. Moreover, it is possible to define any number of new, resource specific, states, for example so called P-states, in which processor can operate.

**Power consumption model.** The main aim of these models is to emulate the behavior of the real computing resource and the way it consumes power. Due to a rich functionality and flexible environment description, DCworms can be used to verify a number of theoretical assumptions and to develop new power consumption models. Modeling of power consumption is realized by the energy estimation plugin that calculates energy usage based on information about the resource power profile, resource utilization, and the application profile including energy consumption and heat production metrics. Relation between model and power profile is illustrated in Figure 3.

**Power management interface.** DCworms is complemented with an interface that allows scheduling plugins to collect detailed power information about computing resource components and to change their power states. It enables performing various operations on the given resources, including dynamically changing the frequency level of a single processor, turning off/on computing resources etc. The activities performed with this interface find a reflection in total amount of energy consumed by the resource during simulation.

Presence of detailed resource usage information, current resource energy state description and functional energy management interface enables an im-

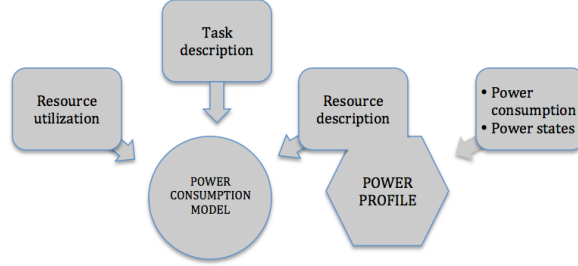


Figure 3: Power consumption modeling

plementation of energy-aware scheduling algorithms. Resource energy consumption becomes in this context an additional criterion in the scheduling process, which uses various techniques to decrease energy consumption, e.g. workload consolidation, moving tasks between resources to reduce a number of running resources, dynamic power management, cutting down CPU frequency, and others.

### 3.5. Application performance modeling

In general, DCworms implements user application models as objects describing computational, communicational as well as energy requirements and profiles of the task to be scheduled. Additionally, simulator provides means to include complex and specific application performance models during simulations. They allow researchers to introduce specific ways of calculating task execution time. These models can be plugged into the simulation environment through a dedicated API and implementation of an appropriate plugin. To specify the execution time of a task user can apply a number of parameters, including:

- task length (number of CPU instructions)
- task requirements
- detailed description of allocated resources (processor type and parameters, available memory)
- input data size
- network parameters

Using these parameters developers can for instance take into account the architectures of the underlying systems, such as multi-core processors, and their impact on the final performance of applications.

#### 4. Modeling of energy consumption in DCworms

DCworms is an open framework in which various models and algorithms can be investigated as presented in Section 3.5. In this section, we discuss possible approaches to modeling that can be applied to simulation of energy-efficiency of distributed computing systems. In general, to facilitate the simulation process, DCworms provides some basic implementation of power consumption, air throughput and thermal models. We introduce power consumption models as examples and validate part of them by experiments in real computing system (in Section 5). Description of thermal models and corresponding experiments was presented in [2].

The most common questions explored by researchers who study energy-efficiency of distributed computing systems is how much energy  $E$  do these systems require to execute workloads. In order to obtain this value the simulator must calculate values of power  $P_i(t)$  and load  $L_i(t)$  in time for all  $m$  computing nodes,  $i = 1..m$ . Load function may depend on specific load models applied. In more complex cases it can even be defined as vectors of different resource usage in time. In a simple case load can be either idle or busy but even in this case estimation of job processing times  $p_j$  is needed to calculate total energy consumption. The total energy consumption of computing nodes is given by (1):

$$E = \sum_i^m \int_t P_i(t) dt \quad (1)$$

Power function may depend on load and states of resources or even specific applications as explained in Section 4.1. Total energy can be also completed by adding constant power usage of components that does not depend on load or state of resources.

In large computing systems which are often characterized by high computational density, total energy consumption of computing nodes is not the only result interesting for researchers. Temperature distribution is getting more and more important as it affects the energy consumption of cooling devices, which can reach even half of a total data center energy use. In

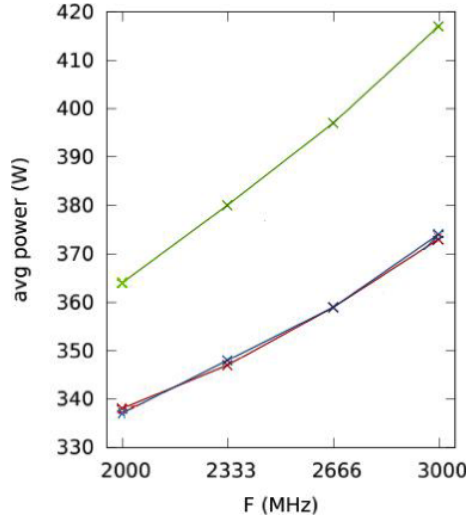


Figure 4: Average power usage with regard to CPU frequency - Linpack (*green*), Abinit (*purple*), Namd (*blue*) and Cpuburn (*red*).

order to obtain accurate values of temperatures heat transfer simulations based on the Computational Fluid Dynamics (CFD) methods have to be performed. These methods require as an input (i.e. boundary conditions) a heat dissipated by IT hardware and air throughput generated by fans at servers' outlets. Another approach is based on simplified thermal models that without costly CFD calculations provide rough estimations of temperatures. DCworms enables the use of both approaches. In the former, the output of simulations including power usage of computing nodes in time and air throughput at node outlets can be passed to CFD solver. Details addressing this integration issues are introduced in [15].

#### 4.1. Power consumption models

As stated above power usage of computing nodes depend on a number of factors.

Generally, the power consumption of a modern CPU is given by the formula:

$$P = C \cdot V_{core}^2 \cdot f \quad (2)$$

with  $C$  being the processor switching capacitance,  $V_{core}$  the current P-State's core voltage and  $f$  the frequency. Based on the above equation it

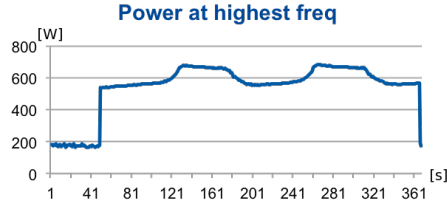


Figure 5: Power in time for the highest frequency

is suggested that although the reduction of frequency causes an increase in the time of execution, the reduction of frequency also leads to the reduction of  $V_{core}$  and thus the power savings from the  $P \sim V_{core}^2$  relation outweigh the increased computation time. However, experiments performed on several HPC servers show that this dependency does not reflect theoretical shape and is often close to linear as presented in Figure 4. This phenomenon can be explained by impact of other component than CPU and narrow range of available voltages. A good example of impact by other components is power usage of servers with visible influence of fans as illustrated in Figure 5.

For these reasons, DCworms allows users to define dependencies between power usage and resource states (such as CPU frequency) in the form of tables or arbitrary functions using energy estimation plugins.

The energy consumption models provided by default can be classified into the following groups, starting from the simplest model up to the more complex ones. Users can easily switch between the given models and incorporate new, visionary scenarios.

#### 4.2. Static approach

Static approach is based on a static definition of resource power usage. This model calculates the total amount of energy consumed by the computing resource system as a sum of energy, consumed by all its components (processors, disks, power adapters, etc.). More advanced versions of this approach assume definition of resource states along with corresponding power usage. This model follows changes of resource power states and sums up the amounts of energy defined for each state. In this case, specific values of power usage are defined for all discrete  $n$  states as shown in (3):

$$S_1 \rightarrow P_1, S_2 \rightarrow P_2, \dots, S_n \rightarrow P_n \quad (3)$$

Within DCworms we built in a static approach model that uses common resource states that affect power usage which are the CPU power states. Hence, with each node power state, understood as a possible operating state (p-state), we associated a power consumption value that derives from the averaged values of measurements obtained for different types of application. We distinguish also an idle state. Therefore, the current power usage of the node can be expressed as:  $P = P_{idle} + P_f$  where  $P$  denotes power consumed by the node,  $P_{idle}$  is a power usage of node in idle state and  $P_f$  stands for power usage of CPU operating at the given frequency level. Additionally, node power states are taken into account to reflect no (or limited) power usage when a node is off.

#### 4.3. Resource load

Resource load model extends the static power state description and enhances it with real-time resource usage, most often simply the processor load. In this way it enables a dynamic estimation of power usage based on resource basic power usage and state (defined by the static resource description) as well as resource load. For instance, it allows distinguishing between the amount of energy used by idle processors and processors at full load. In this manner, energy consumption is directly connected with power state and describes average power usage by the resource working in a current state. In this case, specific values of power usage are defined for all pairs state and load values (discretized to  $l$  values) as shown in (4):

$$(S_1, L_1) \rightarrow P_{11}, (S_1, L_2) \rightarrow P_{12}, \dots, (S_2, L_1) \rightarrow P_{21}, \dots, (S_n, L_l) \rightarrow P_{nl}, \quad (4)$$

A typical functional model of power usage can be based on theoretical dependencies between power and parameters such as CPU frequency, voltage, load, memory usage, etc. In this case CPU power usage for core  $i$ ,  $P_i$  can be given according to (2). Then, the total CPU power can be calculated as a sum of utilized cores:

$$P_{CPU} = P_{idle} + \sum_i^{num\_cores} P_i \quad (5)$$

and the whole node power usage as a sum of CPU, memory and other components (e.g., as defined in [14]):

$$P = \sum_i^{num\_cpus} P_{CPU_i} + P_{RAM} + P_{HD} + P_{fan} + P_{PSU} \quad (6)$$

Unfortunately, to verify this model and adjust it to the specific hardware, power usage of particular subcomponents such as CPU or memory must be measured. As this is usually difficult, other models, based on a total power use, can be applied.

An example is a model applied in DCworms based on the real measurements (see Section 5.3 for more details):

$$P = P_{idle} + L * P_{cpubase} * c^{(f-f_{base})/100} + P_{app}, \quad (7)$$

where  $P$  denotes power consumed by the node executing the given application,  $P_{idle}$  is a power usage of node in idle state,  $L$  is the current utilization level of the node,  $P_{cpubase}$  stands for power usage of fully loaded CPU working in the lowest frequency,  $c$  is the constant factor indicating the increase of power consumption with respect to the frequency increase  $f$  is a current frequency,  $f_{base}$  is the lowest available frequency within the given CPU and  $P_{app}$  denotes the additional power usage derived from executing a particular application ( $P_{app}$  is a constant appointed experimentally for each application in order to extract the part of power consumption independent of the load and specific for particular type of task).

#### 4.4. Application specific

Application specific model allows expressing differences in the amount of energy required for executing various types of applications at diverse computing resources. It considers all defined system elements (processors, memory, disk, etc.), which are significant in total energy consumption. Moreover, it also assumes that each of these components can be utilized in a different way during the experiment and thus have different impact on total energy consumption. To this end, specific characteristics of resources and applications are taken into consideration. Various approaches are possible including making the estimated power usage dependent on defined classes of applications, ratio between CPU-bound and IO-bound operations, etc. In this case, power usage is an arbitrary function of state, load, and application characteristics as shown in (8):

$$f(S, L, A) \rightarrow P \quad (8)$$



## 5. Experiments and evaluation

In this section, we present computational analysis that were conducted to emphasize the role of modeling and simulation in studying computing systems performance. To this end we evaluate the impact of energy-aware resource management policies on overall energy-efficiency of specific workloads on heterogeneous resources. The following sections contain description of the used system, tested application and the results of simulation experiments conducted for the evaluated strategies.

### 5.1. Testbed description

To obtain values of power consumption that could be later used in DC-worms environment to build the model and to evaluate resource management policies we ran a set of applications / benchmarks on the physical testbed. For experimental purposes we choose the Christmann high-density Resource Efficient Cluster Server (RECS) system [3]. The single RECS unit consists of 18 single CPU modules, each of them can be treated as an individual node of PC class. Configuration of our RECS unit is presented in Table 1.

Nodes		
Type	Memory (RAM)	Count
Intel i7	16 GB	8
AMD Fusion T40N 64 Bit	4 GB	6
Atom D510 64 Bit	2 GB	4

Table 1: RECS system configuration

The RECS system was chosen due to its heterogeneous platform with very high density and energy efficiency that has a monitoring and controlling mechanism integrated. The built-in and additional sensors allow monitoring the complete testbed at a very fine granularity level without the negative impact of the computing- and network-resources.

### 5.2. Evaluated applications

As mentioned, first we carried out a set of tests on the real hardware used as a CoolEmAll testbed to build the performance and energy profiles of applications. The following applications were taken into account:

**Abinit** [16] is a widely-used application for computational physics simulating systems made of electrons and nuclei to be calculated within density functional theory.

**C-Ray** [17] is a ray-tracing benchmark that stresses floating point performance of a CPU. Our test is configured with the 'scene' file at 16000x9000 resolution.

**Linpack** [19] benchmark is used to evaluate system floating point performance. It is based on the Gaussian elimination methods that solve a dense  $N$  by  $N$  system of linear equations.

**Tar** [20] it is a widely used data archiving software. In our tests the task was to create one compressed file of Linux kernel (version 3.4), which is about 2.3 GB size, using bzip2.

**FFTE** [18] benchmark measures the floating-point arithmetic rate of double precision complex one-dimensional Discrete Fourier Transforms of 1-, 2-, and 3-dimensional sequences of length  $2^p * 3^q * 5^r$ . In our tests only one core was used to run the application.

### 5.3. Models

Based on the measured values we evaluated three types of models that can be applied, among others, to the simulation environment.

**Static** This model refers to the static approach presented in Section 4.1. According to the measured values we created a resource power consumption model that is based on a static definition of resource power usage.

**Dynamic** This model refers to the Resource load approach presented in Section 4.1. Based on the measured values of the total node power usage for various levels of load and frequencies of CPUs node power usage was defined as in 7.

Table 2 and Table 3 contain values of  $P_{cpubase}$  and  $P_{app}$ , respectively, obtained for the particular application and resource architectures. Lack of the corresponding value means that the application did not run on the given type of node.

Intel I7	AMD Fusion	Atom D510
8	2	1

Table 2:  $P_{cpubase}$  values in Watts

**Mapping** This model refers to the Application specific approach presented in Section 4.1. However, in this model we applied the measured values

Application	Node type		
	Intel I7	AMD Fusion	Atom D510
Abinit	3.3	-	-
Linpactiny	2.5	-	0.2
Linpact3gb	6	-	-
C-Ray	4	1	0.05
FFT	3.5	2	0.1
Tar	3	2.5	0.5

Table 3:  $P_{app}$  values in Watts

for each application exactly to the power model. Neither dependencies with load nor application profiles are modeled. Obviously this model is contaminated only with the inaccuracy of the measurements and variability of power usage (caused by other unmeasured factors).

The following table (Table 4) contains the relative errors of the models with respect to the measured values

Static	Dynamic	Mapping
13.74	10.85	0

Table 4: Power models error in %

Obviously, 0% error in the case of the Mapping model is caused by the use of a tabular data, which for each application stores a specific power usage. Nevertheless, in all models we face possible deviations from the average caused by power usage fluctuations not explained by variables used in models. These deviations reached around 7% for each case.

For the experimental purposes we decided to use the latter model. Thus, we introduce into the simulation environment exact values obtained within our testbed, to build both the power profiles of applications as well as the application performance models, denoting their execution times.

#### 5.4. Methodology

Every chosen application / benchmark was executed on each type of node, for all frequencies supported by the CPU and for different levels of parallelization (number of cores). To eliminate the problem with assessing which part of the power consumption comes from which application, in case when more then one application is ran on the node, the queuing system (SLURM)

was configured to run jobs in exclusive mode (one job per node). Such configuration is often used for at least dedicated part of HPC resources. The advantage of the exclusive mode scheduling policy consists in that the job gets all the resources of the assigned nodes for optimal parallel performance and applications running on the same node do not influence each other. For every configuration of application, type of node and CPU frequency we measure the average power consumption of the node and the execution time. The aforementioned values were used to configure the DCworms environment providing energy and time execution models. Based on the models obtained for the considered set of resources and applications we evaluated a set of resource management strategies in terms of energy consumption needed to execute four workloads varying in load intensity (10%, 30%, 50%, 70%). The differences in the load were obtained by applying various intervals (3000, 1200, 720 and 520 seconds, respectively) related to submission times of two successive tasks. In all cases the number of tasks was equal to 1000. Moreover, we differentiated the applications in terms of number of cores allocated by them and their type. To generate a workload we used the DCworms workload generator tool with the aforementioned characteristics gathered in Table 5.

In all cases we assumed that tasks are scheduled and served in order of their arrival (FIFO strategy) using relaxed backfilling approach, with indefinite delay for the highest priority task. Moreover, all tasks were assigned to nodes with the condition that they can be assigned only to nodes of the type on which the application was able to run (in other words - we had the corresponding value of power consumption and execution time).

### *5.5. Computational analysis*

In the following section we present the results obtained for the workload with load density equal to 70% in the light of five resource management and scheduling strategies. The scheduling strategies were evaluated according to two criteria: total energy consumption and maximum completion time of all tasks (makespan). These evaluation criteria employed in our experiments represent interests of various groups of stakeholders present in data centers. Then we discuss the corresponding results received for workloads with other density level.

Characteristic	Load intensity				Distribution
	10	30	50	70	
Task Count	1000				constant
Task Interval [s]	3000	1200	720	520	poisson
Number of cores to run	1				uniform - 30%
	2				uniform - 30%
	3				uniform - 10%
	4				uniform - 10%
	5				uniform - 5%
	6				uniform - 5%
	7				uniform - 5%
	8				uniform - 5%
Application type	Abinit				uniform - 20%
	C-Ray				uniform - 20%
	Tar				uniform - 20%
	Linpack - 3Gb				uniform - 10%
	Linpack - tiny				uniform - 10%
	FFT				uniform - 20%

Table 5: Workload characteristics

#### 5.5.1. Random approach

The first considered by us policy was the Random (R) strategy in which tasks were assigned to nodes in a random manner. The Random strategy is only the reference one and will be later used to compare benefits in terms of energy efficiency resulting from more sophisticated algorithms. Criteria values are as follows: **total energy usage**: 46.883 kWh, **workload completion time**: 533 820 s.

In the second version of this strategy, which is getting more popular due to energy costs, we switched off unused nodes to reduce the total energy consumption. In the previous one, unused nodes are not switched off, which case is still the primary one in many HPC centers. In this version of experiment we neglected additional cost and time necessary to change the power state of resources. As can be observed in the Figure 6, switching off unused nodes led to decrease of the total energy consumption.

As expected, with respect to the makespan criterion, both approaches perform equally reaching **workload completion time**: 533 820 s. However, the pure random strategy was significantly outperformed in terms of energy



Figure 6: Comparison of energy usage for Random (left) and Random + switching off unused nodes strategy (right)

usage, by the policy with additional node power management with its **total energy usage**: 36.705 kWh. The overall energy savings reached 22%.

### 5.5.2. Energy optimization

The next two evaluated resource management strategies try to decrease the total energy consumption (EO) caused by the execution of the whole workload. They take into account differences in applications and hardware profiles by trying to find the most energy efficient assignment. In the first case we assumed that there is again no possibility to switch off unused nodes, thus for the whole time needed to execute workload nodes consume at least power for idle state. To obtain the minimal energy consumption, tasks have to be assigned to the nodes for which the difference between energy usage for the node running the application and the node in the idle state is minimal. The power consumption measured in idle state for three types of nodes is gathered in the Table 6.

Type of processor within the node	Power usage in idle state [W]
Intel i7	11.5
AMD Fusion	10
Atom D510	19

Table 6: Measured power of testbed nodes in idle state

As mentioned, we assign tasks to nodes minimizing the value of expression:  $(P - P_{idle}) * exec\_time$ , where  $P$  denotes observed power of the node

running the particular application and *exec.time* refers to the measured application running time. Based on the application and hardware profiles, we expected that Atom D510 would be the preferred node. Obtained schedule, that is presented in the Gantt chart in Figure 7 confirmed our assumptions. Atom D510 nodes are nearly fully loaded, while the least energy-favourable AMD nodes are used only when other ones are busy.

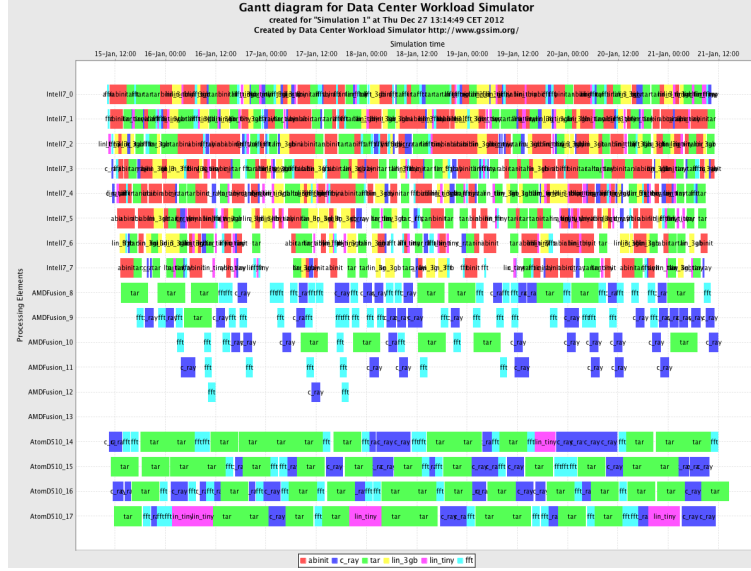


Figure 7: Energy usage optimization strategy

This allocation strategy, leads to slight deterioration of makespan criterion, resulting in **workload completion time** equal to 534 400 s. Nevertheless, the **total energy usage** is reduced, achieving: 46.305 kWh.

The next strategy is similar to the previous one, so making the assignment of task to the node, we still take into consideration application and hardware profiles, but in that case we assume that the system supports possibility of switching off unused nodes. In this case the minimal energy consumption is achieved by assigning the task to the node for which the product of power consumption and time of execution is minimal. In other words we minimized the following expression:  $P * exec.time$ . Contrary to the previous strategy, we expected Intel I7 nodes to be allocated first. Generated Gantt chart is consistent with our expectations.

Estimated **total energy usage** of the system is 30.568 kWh. As we can see, this approach significantly improved the value of this criterion, com-

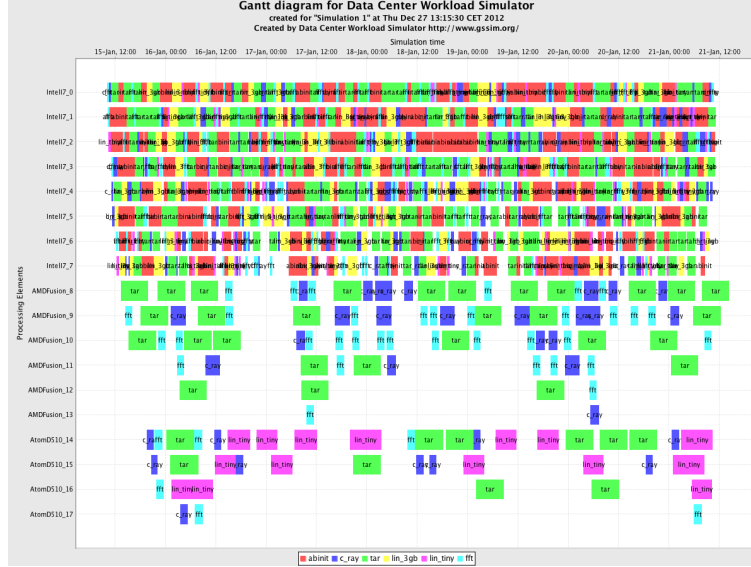


Figure 8: Energy usage optimization + switching off unused nodes strategy

paring to the previous policies. Moreover, the proposed allocation strategy does not worsen the **workload completion time** criterion, for which the resulting value is equal to 533 820 s.

### 5.5.3. Downgrading frequency

The last case considered by us is modification of the random strategy. We assume that tasks do not have deadlines and the only criterion which is taken into consideration, is the total energy consumption. In this experiment we configured the simulated infrastructure for the lowest possible frequencies of CPUs (LF). The experiment was intended to check if the benefit of running the workload on less power-consuming frequency of CPU is not leveled by the prolonged time of execution of the workload. The values of the evaluated criteria are as follows: **workload completion time**: 1 065 356 s and **total energy usage**: 77.109 kWh. As we can see, for the given load of the system (70%), the cost of running the workload that requires almost twice more time, can not be compensate by the lower power draw. Moreover, as it can be observed on the chart in Figure 9, the execution times on the slowest nodes (Atom D510) visibly exceeds the corresponding values on other servers.

As we were looking for the trade-off between total completion time and energy usage, we were searching for the workload load level that can benefit



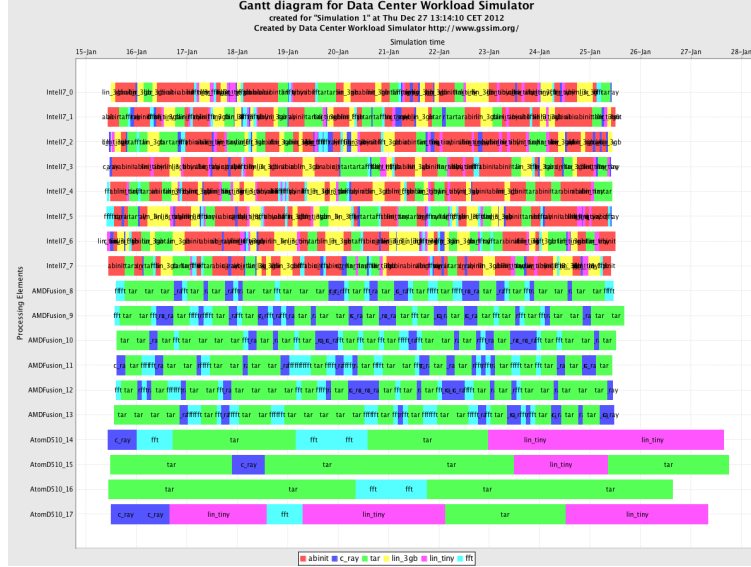


Figure 9: Frequency downgrading strategy

from the lower system performance in terms of energy-efficiency. For the frequency downgrading policy, we noticed the improvement on the energy usage criterion only for the workload resulting in 10% system load. For this threshold we observed that slowdown in task execution does not affect the subsequent tasks in the system and thus the total completion time of the whole workload.

Figure 10 shows schedules obtained for Random and Random + lowest frequency strategy.

### 5.6. Discussion

The following tables: Table 7 and Table 8 contain the values of evaluation criteria (total energy usage and makespan respectively) gathered for all investigated workloads.

Referring to the Table 7, one should easily note that gain from switching off unused nodes decreases with the increasing workload density. In general, for the highly loaded system such policy does not find an application due to the cost related to this process and relatively small benefits. Another interesting conclusion, refers to the poor result for Random strategy with downgrading the frequency approach. The lack of improvement on the energy usage criterion for higher system load can be explained by the rel-

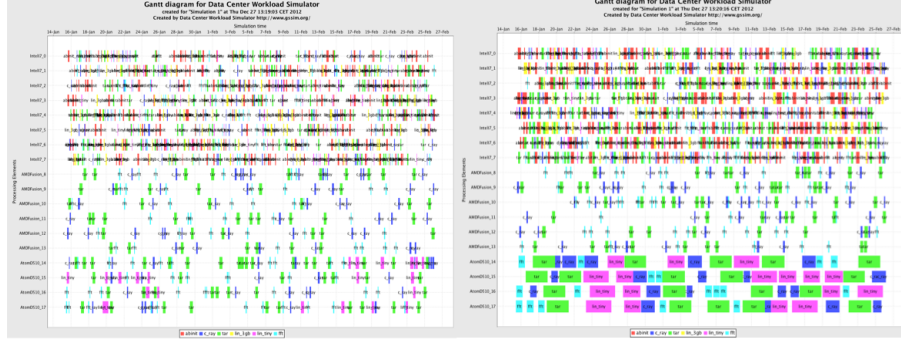


Figure 10: Schedules obtained for Random strategy (left) and Random + lowest frequency strategy (right) for 10% of system load

Load	Strategy				
	R	R+NPM	EO	EO+NPM	R+LF
10%	241.337	37.811	239.667	25.571	239.278
30%	89.853	38.059	88.823	25.595	90.545
50%	59.112	36.797	58.524	26.328	76.020
70%	46.883	36.705	46.305	30.568	77.109

Table 7: Energy usage [kWh] for different level of system load. R - Random, R+NPM - Random + node power management, EO - Energy optimization, EO+NPM - Energy optimization + node power management, R+LF - Random + lowest frequency

Load	Strategy				
	R	R+NPM	EO	EO+NPM	R+LF
10%	3 605 428	3 605 428	3 605 428	3 605 428	3 622 968
30%	1 214 464	1 214 464	1 215 044	1 200 807	1 275 093
50%	729 066	729 066	731 126	721 617	1 049 485
70%	533 820	533 820	534 400	533 820	1 065 356

Table 8: Makespan [s] for different level of system load. R - Random, R+NPM - Random + node power management, EO - Energy optimization, EO+NPM - Energy optimization + node power management, R+LF - Random + lowest frequency

atively small or no benefit obtained for prolonging the task execution, and thus, maintaining the node in working state. The cost of longer workload completion can not be compensate by the very little energy savings derived from the lower operating state of node. The greater criteria values for the higher system load are the result of greater time space between submission

of successive tasks, and thus longer workload execution. Based on Table 8, one should note that differences in workload completion times are relatively small for all evaluated policies, except Random + lowest frequency approach.

We also demonstrated differences between power usage models. They span from rough static approach to accurate application specific models. However, the latter can be difficult or even infeasible to use, as it requires real measurements for specific applications beforehand. This issue can be partially resolved by introducing application profiles and classification, which can deteriorate the accuracy though. This issue is begin studied more deeply within CoolEmAll project.

## 6. Conclusions and future work

In this paper we presented a Data Center Workload and Resource Management Simulator (DCworms) which enables modeling and simulation of computing infrastructures to estimate their performance, energy consumption, and energy-efficiency metrics for diverse workloads and management policies. DCworms provides broad options of customization and combines detailed applications and workloads modeling with simulation of data center resources including their power usage and thermal behavior. We shown its energy-efficiency related features and proposed three methods of power usage modeling: static (fully defined by resource state), dynamic (defined by a function of parameters such as CPU frequency and load), and mapping (based on power usage of specific applications). We compared results of simulations to measurements of real servers and shown differences in accuracy and usability of these models. We also demonstrated DCworms capabilities to implement various resource management policies including workload scheduling and node power management. The experimental studies we conducted shown that their impact on overall energy-efficiency depends on a type of servers, their power usage in idle time, possibility of switching off nodes as well as level of load. DCworms is a part of the Simulation, Visualisation and Decision Support (SVD) Toolkit being developed within the CoolEmAll project. The aim of this toolkit is, based on data center building blocks defined by the project, to analyze energy-efficiency of data centers taking into account various aspects such as heterogenous hardware architectures, applications, management policies, and cooling. DCworms will take as an input the open models of the data center building blocks and application profiles. DCworms will be applied to evaluation of resource management approaches.

These policies may include a wide spectrum of energy-aware strategies such as workload consolidation, dynamic switching off nodes, DVFS and thermal-aware methods. Output of simulations will include distribution of servers' power usage in time along with estimations of server outlets air flow. These data will be passed to Computational Fluid Dynamics (CFD) simulations using OpenFOAM solver and to advanced 3D visualization. In this way users will be able to study energy-efficiency of a data center from a detailed analysis of workloads and policies to the impact on heat transfer and overall energy consumption. Thus, future work on DCworms will focus on more precise power, air-throughput, and thermal models. Additional research directions will include modeling application execution phases, adding predefined common HPC and cloud management policies and application performance and resource power models.

## Acknowledgement

The results presented in this paper are partially funded by the European Commission under contract 288701 through the project CoolEmAll and by grants from Polish National Science Center: a grant under award number 636/N-COST/09/2010/0 and a grant under award number 5790/B/T02/2010/38.

## References

- [1] A. Berl, E. Gelenbe, M. di Girolamo, G. Giuliani, H. de Meer, M.-Q. Dang, K. Pentikousis. Energy-Efficient Cloud Computing. *The Computer Journal*, 53(7), 2010.
- [2] M. vor dem Berge, G. Da Costa, M. Jarus, A. Oleksiak, W. Piatek, E. Volk. Modeling Data Center Building Blocks for Energy-efficiency and Thermal Simulations. 2nd International Workshop on Energy-Efficient Data Centres, Berkeley, 2013.
- [3] M. vor dem Berge, G. Da Costa, A. Kopecki, A. Oleksiak, J-M. Pierson, T. Piontek, E. Volk, S. Wesner. Modeling and Simulation of Data Center Energy-Efficiency in CoolEmAll. *Energy Efficient Data Centers, Lecture Notes in Computer Science Volume 7396*, 2012, pp 25-36.
- [4] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, R. Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms,

Software: Practice and Experience (SPE), Volume 41, Number 1, Pages: 23-50, ISSN: 0038-0644, Wiley Press, New York, USA, January, 2011.

- [5] <http://dcs.bcs.org/welcome-dcsg-simulator>
- [6] <http://www.datacenterdynamics.com/blogs/ian-bitterlin/it-does-more-it-says-tin%E2%80%A6>
- [7] E. Gelenbe, C. Morfopoulou. Power savings in packet networks via optimised routing. *Mobile Networks and Applications*, 17(1):152159, February 2012.
- [8] G. L. T. Chetsa, L. Lefèvre, J-M. Pierson, P. Stolf, G. Da Costa. DNA-inspired Scheme for Building the Energy Profile of HPC Systems. In: *International Workshop on Energy-Efficient Data Centres*, Madrid, Springer, 2012.
- [9] A. Kipp, L. Schubert, J. Liu, T. Jiang, W. Christmann, M. vor dem Berge. Energy Consumption Optimisation in HPC Service Centres, *Proceedings of the Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, B.H.V. Topping and P. Iványi, (Editors), Civil-Comp Press, Stirlingshire, Scotland, 2011.
- [10] D. Kliazovich, P. Bouvry, and S. U. Khan, A Packet-level Simulator of Energy-aware Cloud Computing Data Centers, *Journal of Supercomputing*, vol. 62, no. 3, pp. 1263-1283, 2012.
- [11] S. Bak, M. Krystek, K. Kurowski, A. Oleksiak, W. Piatek and J. Weglarz, GSSIM - a Tool for Distributed Computing Experiments, *Scientific Programming Journal*, vol. 19, no. 4, pp. 231-251, 2011.
- [12] M. Krystek, K. Kurowski, A. Oleksiak, W. Piatek, Energy-aware simulations with GSSIM. *Proceedings of the COST Action IC0804 on Energy Efficiency in Large Scale Distributed Systems*, 2010, pp. 55-58.
- [13] J. Koomey. 2008. Worldwide electricity used in data centers. *Environmental Research Letters*. vol. 3, no. 034008. September 23.
- [14] O. Mämmelä, M. Majanen, R. Basmadjian, H. De Meer, A. Giesler, W. Homberg, Energy-aware job scheduler for high-performance computing, *Computer Science - Research and Development*, November 2012, Volume 27, Issue 4, pp 265-275.

- [15] U. Woessner, E. Volk, G. Gallizo, M. vor dem Berge, G. Da Costa, P. Domagalski, W. Piatek, J-M. Pierson. (2012) D2.2 Design of the CoolEmAll simulation and visualisation environment - CoolEmAll Deliverable, <http://coolemall.eu>
- [16] Abinit. <http://www.abinit.org/>, 2013.
- [17] C-ray ray-tracing benchmark. <http://code.google.com/p/cray/>, 2013.
- [18] FFTW: A Fast Fourier Transform Package. <http://www.fftw.jp/>, 2013.
- [19] Linpack. <http://www.netlib.org/linpack/>, 2013.
- [20] Tar data archiving software. <http://www.gnu.org/software/tar/>, 2013.
- [21] <http://gwa.ewi.tudelft.nl/>
- [22] <https://computing.llnl.gov/linux/slurm/>
- [23] Parallel Workload Archive, <http://www.cs.huji.ac.il/labs/parallel/workload/>
- [24] <http://www.adaptivecomputing.com/products/open-source/torque/>
- [25] Colt Modular Data Centre, <http://www.colt.net/uk/en/products-services/data-centre-services/modular-data-centre-en.htm>
- [26] The CoolEmAll project website, <http://coolemall.eu>
- [27] EcoCooling, <http://www.ecocooling.org>
- [28] Future Facilities, <http://www.futurefacilities.com/>
- [29] The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE, <http://www.thegreengrid.org/Global/Content/white-papers/The-Green-Grid-Data-Center-Power-Efficiency-Metrics-PUE-and-DCiE>
- [30] SGI ICE Cube Air, [http://www.sgi.com/products/data\\_center/ice\\_cube\\_air/](http://www.sgi.com/products/data_center/ice_cube_air/)