Energy aware ultrascale systems

Ariel Oleksiak, Laurent Lefèvre, Pedro Alonso, Georges da Costa, Vincenzo de Maio, Neki Frasheri, Victor Garcia, Joel Guerrero, Sébastien Lafond, Alexey Lastovetsky, et al.

Chapter 5

ENERGY AWARE ULTRASCALE SYSTEMS

A. Oleksiak[1] Laurent Lefevre[2] and Pedro Alonso [3] and Georges Da Costa[4] and Vincenzo De Maio[5] and Neki Frasheri[6] and V. M. Garcia [7] and Joel Guerrero[8] and Sebastien Lafond[9] and Alexey Lastovetsky [10] and Ravi Reddy Manumachu [11] and Benson Muite[12] and Anne-Cecile Orgerie[13] and Wojciech Piatek[14] and Jean-Marc Pierson[15] and Radu Prodan[16] and Patricia Stolf[17] and Enida Sheme[18] and Sebastien Varrette[19]

Keywords: sustainability, energy efficiency metrics, energy models, energy monitoring, energy simulation, energy aware software, energy consumption, heterogeneous systems.

Energy consumption is one of the main limiting factor for the design of Ultrascale infrastructures . Multi-level hardware and software optimizations must be designed

[1]Poznan Supercomputing and Networking Center, Poland
[2]Ecole Normal Superior, Lyon, France
[3]Universitat Politecnica de Valencia, Spain
[4]University of Toulouse, France
[5]Vienna University of Technology, Austria
[6]Polytechnic University of Tirana, Albania
[7]Universitat Politecnica de Valencia, Spain
[8]University of Genoa, Italy
[9]Abo Akademi University, Finland
[10]University College of Dublin, Ireland
[11]University College of Dublin, Ireland
[12]Institute of Computer Science, Tartu Ulikool, Estonia
[13]Univ. Rennes, Inria, CNRS, IRISA, France
[14]Poznan Supercomputing and Networking Center, Poland
[15]University of Toulouse, France
[16]Alpen-Adria Universitat Klagenfurt, Austria
[17]University of Toulouse
[18]Polytechnic University of Tirana, Albania
[19]University of Luxembourg, Luxembourg

and explored in order to reduce energy consumption for these large scale equipment. This chapter addresses the issue of energy efficiency of Ultrascale Systems in front of other quality metrics. The goal of this chapter is to explore the design of metrics, analysis, frameworks and tools for putting energy awareness and energy efficiency at the next stage. Significant emphasis will be placed on the idea of "energy complexity", reflecting the synergies between energy efficiency and Quality of Service, resilience and performance, by studying computation power, communication/data sharing power, data access power, algorithm energy consumption, etc.

5.1 Energy modeling and simulation

Energy consumption has become a significant issue for data centers. For this reason, many researchers currently focus on developing energy aware algorithms to improve their energy efficiency. However, due to the difficulty of employing real data centers' infrastructure for assessing the effectiveness of energy-aware algorithms, researchers resort on simulation tools. These tools require precise and detailed models for virtualized data centers in order to deliver accurate results. In recent years, many models have been proposed, but most of them do not consider some of the energy impacting components (e.g. CPU, network, storage). In this work, we provide a review of energy modeling and simulations for Ultrascale systems and identify open issues and future challenges in the field.

First, we focus on energy consumption of physical nodes inside the data center. Since energy is the integral of power over time, those are the two main factors that are interested by energy modeling. Therefore, first we provide models for the component with the highest power draw inside the physical nodes, like CPU, RAM, I/O, network and cooling systems. After this, we focus on modeling the parameters affecting the running time of the data center, such as the data center's workload and the way computational resources are managed inside the data center.

Concerning the simulation part, we discuss different types of data center simulations, such as event-based simulation and statistical simulations, discussing the current state of the art in this area and the solutions that can fit the most the Ultrascale scenario.

5.1.1 Related work

Global power and energy models of servers inside data centers are evolving. In [196], authors provide a global landscape of power consumption of servers il large scale data centers and show the relative importance of each subpart: (e.g. CPU, memory, network,...).

Concerning CPU modeling, existing papers either focus on a specific CPU architecture [197, 198, 47] or assume a linear relationship between CPU usage and energy consumption [199, 200], which may lead to inaccurate results [201]. Moreover, most of this models do not consider the virtualization overhead, making it not suitable for virtualized data centre.

Concerning memory modeling, works like [202, 203] provide simulations of DRAM behaviour, but neither the virtualization overhead nor the energy consumption is considered. Works like [204] provides modeling of memory resource management in VMWare ESX Server, but no energy modeling is provided. Similar works like [205, 206] provide insights about memory management, respectively for Xen and KVM hypervisor.

Storage energy modeling has been provided by [207] and other similar works like [208, 209, 210]. These works, however, do not consider the virtualization overhead, neither distributed storage systems.

Data centres network performances has been instead modeled in works like [211]. Other works like [212] propose data center network simulators, without considering energy consumption. Works like [213, 214] target how to improve energy efficiency of networking in data centres, while works like [215, 216] try to model energy consumption of network transfers, but do not use this model in simulations.

Works like [217, 218, 219] provide a joint approach to data center modeling. However, all these works have the problems previously outlined. Moreover several features provided by the virtualization layer available in data centers are not modeled. We analyze them in details in the following sections.

5.1.2 Actor definition

First of all, we define the actors involved in energy consumption. We define a Ultrascale system $\mathscr{S}$ as:

Definition 1. *A **ultrascale system** is a vector $\mathscr{S}$,*

$$\mathscr{S} = [\mathscr{H}, \mathscr{N}], \tag{5.1}$$

where:

- *$\mathscr{H}$ is the set of physical machines (PM)s inside the system $\mathscr{S}$ during its whole lifetime. PMs are defined in Definition 2;*
- *$\mathscr{N}$ is the network infrastructure, composed by network switches connecting PMs. Network switches are defined in Definition 3.*

A PM is a hardware device, such as a personal computer or any other computing device. PMs can also be used used to host virtual machines (VM)s that execute computation. Resources offered by the PMs are shared between all the VMs that are hosted by the PM. The sharing of the resources is managed by the hypervisor, a program running on the PM operating system (OS), or even a modified version of a PM's OS (e.g. Xen), that is responsible for allowing the different OSs running on the VMs to use the PM's resources. A PM can have a diverse amount of hardware resources. In this work we mostly focus on CPU, RAM, storage and network. For this reason, a PM $h \in \mathscr{H}$ can be seen as a vector of its amount of resources and the set of VMs that are allocated on it.

Definition 2. *We define a PM h as follows:*

$$h = [\mathtt{CPU}_{max}(h), \mathtt{RAM}_{max}(h), \mathtt{BW}_{io}^{max}(h), \mathtt{BW}_{net}^{max}(h), s], \tag{5.2}$$

where:

- $\texttt{CPU}_{max}(h)$ *is the maximum CPU load that is possible to allocate to host h;*
- $\texttt{RAM}_{max}(h)$ *is the maximum amount of RAM for host h;*
- $\texttt{BW}_{io}^{max}(h)$ *is the maximum I/O bandwidth;*
- $\texttt{BW}_{net}^{max}(h)$ *is the maximum network bandwidth;*
- *s is the network switch to which h is connected (see Definition 3).*

From now on, $\texttt{CPU}_{max}(h)$ is defined in MIPS (Millions of Instructions Per Second), $\texttt{RAM}_{max}(h)$ in bytes, $\texttt{BW}_{io}^{max}(h)$ and $\texttt{BW}_{net}^{max}(h)$ in bytes per second.

Definition 3. *A **switch** s is defined as:*

$$s = [\texttt{BW}_{net}^{max}(s), \mathscr{H}_{(s)}, \mathscr{N}_{(s)}], \tag{5.3}$$

where:

- $\texttt{BW}_{net}^{max}(s)$ *is the bandwidth of the switch s;*
- $\mathscr{H}_{(s)}$ *is the set of PMs, defining the hosts that are connected to s;*
- $\mathscr{N}_{(s)}$ *is the set of switches, defining the switches that are connected to s.*

Formally, $\exists x \in \mathscr{H}_{(s)} \iff$ *switch s connects x and* $\exists y \in \mathscr{N}_{(s)} \iff$ *switch s connects y. Clearly,* $\mathscr{H}_{(s)} \subset \mathscr{H}$ *and* $\mathscr{N}(s) \subset \mathscr{N}$.

Concerning the networking, We consider that in modern data centers there are many types of LAN technologies available. Such new technologies are different from the typical LAN technologies and offer an higher latency and throughput, to meet the growing needs of modern Cloud applications. Such technologies might need to rely on a different physical layer technology, different from the typical CSMA/CD used by Ethernet. Among them, the most successful one is Infiniband, that is based on the RDMA technology. We will later on describe both technologies in detail.

5.1.3 Energy modeling

.

The computing oriented energy consumption in a Ultrascale system E_s is given by the integral of its instantaneous power draw $P_s(t)$:

$$E_s = \int_{t_{start}}^{t_{end}} P_s(t)\, dt, \tag{5.4}$$

where $[t_{start}, t_{end}]$ is the interval for which energy consumption is calculated, and $P_s(t)$ is the sum of the idle power of data center infrastructure, consisting of the sum of the idle power of all PMs and switches in the system $P_{inf}(\mathscr{S})$ and the instantaneous power $P(h,t)$ of each node h at time instant t:

$$P_s(t) = P_{inf}(\int) + \sum_{h \in \mathscr{H}(t)} P(h,t). \tag{5.5}$$

$P_{inf}(\mathscr{N})$ can be approximated as a constant and includes the idle constants that will be described in the next sections. Concerning instantaneous PM's power consumption, it is the sum of the instantaneous power consumption of each of its components.

While the consumption of some components (e.g. memory, PCI slots, motherboard) is constant over time, consumption of other components (CPU, RAM, I/O operations, network, thermal) varies depending on the load. Therefore, we distinguish two parts, $P_{idle}(h,t)$ and $P_{active}(h,t)$:

$$P(h,t) = P_{idle}(h) + P_{active}(h,t). \tag{5.6}$$

The idle power $P_{idle}(h)$ is a constant representing the consumption required by the machine just to be on. The active power $P_{active}(h,t)$ is instead dependent on the PM subsystems' load. Its value is comprised between 0 and $P_r(h) = P_{max}(h) - P_{idle}(h)$, where $P_r(h)$ is the size of the interval of values in which $P_{active}(h,t)$ is defined. $P_{max}(h)$ is the maximum power consumption on the PM h. For simplicity, it is assumed that P_{active} is influenced mostly by CPU, RAM, network, I/O operations and cooling system, as stated by [220], therefore it is defined as follows:

$$P_{active}(h,t) = P_{cpu}(h,t) + P_{ram}(h,t) + P_{net}(h,t) + P_{io}(h,t) + P_{cool}(h,t). \tag{5.7}$$

Where $P_{cpu}(h,t)$ is the power draw of CPU, $P_{ram}(h,t)$ the power draw of the RAM, P_{net} is the power consumption of network and P_{io} is the power consumption of disk operations, that are the components identified in the definition 2. As energy consumption is the integral of power draw, the active energy consumption of PM h, E_h is obtained in the following way:

$$E_{active}(h) = \int_{t_{start}}^{t_{end}} P_{active}(h,t)\, dt. \tag{5.8}$$

Then, by applying the linearity of the integral, we obtain the equation of the energy consumption of each component:

$$E_x(h) = \int_{t_{start}}^{t_{end}} P_x(h,t)\, dt, \tag{5.9}$$

Where $x \in \{CPU, RAM, net, io, cool\}$.

In the next section, we describe the current state of the art in terms of power modeling for different hardware components, and identify open challenges in adapting such models for Ultrascale machines.

5.1.4 Power modeling

CPU Modeling

CPU is the most impacting component on the energy consumption of a physical machine [221], therefore most work focus only on its model. Most works on energy modeling [40, 222] assume a linear relationship between the CPU use and its power consumption.

However, power consumption is more aligned with a piecewise linear trend according to our observations in Figure 5.1. The initial transition from idle to non-idle state, when several hardware components are simultaneously starting to consume power (e.g. caches), produce in a higher growth in power consumption at low load levels. Once all components are powered up, the power grows at a different trend:

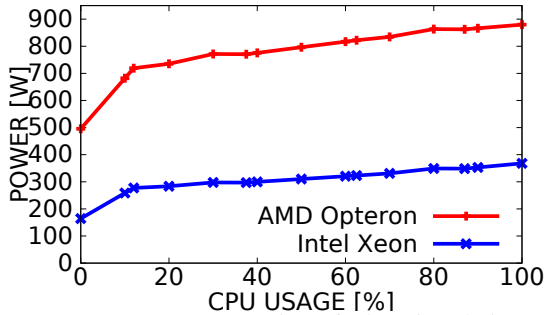


Figure 5.1: Instantaneous power consumption of a host in relation to CPU utilization. Traces coming from the measurements of instantaneous power draw from a AMD Opteron 8356 and a Intel Xeon E5-2690 using a Voltech PM1000+ power measurer.

$$P_{\text{low-cpu}}(h,t) = \alpha(h) \cdot P_r(h) \cdot load_{\text{cpu}}(h,t) \tag{5.10}$$

$$P_{\text{high-cpu}}(h,t) = \beta(h) \cdot P_r(h) + (1-\beta(h)) \cdot P_r(h) \cdot \cdot load_{cpu}(h,t);$$

$$P_{\text{cpu}}(h,t) = \begin{cases} P_{\text{low-cpu}}(h,t), & load_{cpu}(h,t) \leq \mathscr{L}(h); \\ P_{\text{high-cpu}}(h,t), & \text{otherwise}, \end{cases} \tag{5.11}$$

where $\mathscr{L}(h)$ is the load at which the trend changes on host h, $\text{load}_{cpu}(h,t)$ is the CPU load on host h at time instance t, modeled as

$$load_{cpu}(h,t) = \frac{CPU(h,t)}{CPU^{max}(h)} \tag{5.12}$$

$P_{\max}(h)$ and $P_{\text{idle}}(h)$ are the maximum and idle power consumptions of host h, and $\alpha(h)$ and $\beta(h)$ are the coefficients for low (i.e. $\leq \mathscr{L}(h)$) and high (i.e. $> \mathscr{L}(h)$) CPU load levels.

Memory modeling

Workloads for Cloud also include applications such as multimedia, video image processing, and speech recognition, which are extremely memory-intensive. These applications consume a considerable amount of power during memory accesses, that also needs to be modeled. In power modeling for memory, the model described in [223, 224] is employed. Since power consumption varies according to write and read access, we define power function as

$$P_{mem}(h,t) = \beta_{read} access_{read}(h,t) + \beta_{write} access_{write}(h,t) + \mathscr{C}_{memory} \tag{5.13}$$

Where $access_{read}(h,t)$ and $access_{write}(h,t)$ are, respectively, the number of read and write access on the PM h, β_{read} and β_{write} the power consumption for each access (respectively write and read access) and $\mathscr{C}_{memory}$ is the idle power consumption for memory.

Network Modeling
A network transfer occurs in two different directions: sending and receiving. Power draw of the network transfer is different if the transfer is a send or a receive. Therefore, the power draw of network transfer can be modeled as

$$P^x_{net}(h,t) = \delta_x \cdot load_{net}(h,t) + K_x(c_x(t)), \tag{5.14}$$

where $x \in \{s,r\}$ and s,r refer respectively to send and receive. $load_{net}(h)$ models the network load on host h and δ_x the linear relationship between network load and power draw. Concerning the $K_x(c_x(t))$, it models the hardware related constant of the NIC, plus the overhead $O_{net}(c_x(t))$ given by the number of simultaneous connections at time t, $c_x(t)$, formally:

$$\mathscr{K}_x(c_x(t)) = \mathscr{K}_x + O_{net}(c_x(t)), \tag{5.15}$$

When modeling network transfer, further parameters that affects the time t_{end} at which the network transfer ends need to be considered. This time is determined by (1) the network bandwidth on host h, $BW^{max}_{net}(h)$ and (2) the delay caused by transfer patterns, modeled by the parameters b_x and t_x, as in:

$$t_{end} = \frac{\texttt{DATA}_x}{\texttt{BW}^{max}_{net}(h)}, \tag{5.16}$$

$\texttt{DATA}_x$ is the real number of bytes transferred, that is the sum of actual data transferred by the application, plus the additional data needed for routing and transfer control, that is calculated according to

$$\texttt{DATA}_x = \texttt{PAYLOAD}_x + \frac{\texttt{PAYLOAD}_x}{p_x} \cdot \texttt{HEADER}, \tag{5.17}$$

where $\texttt{PAYLOAD}_x$ is the quantity of data to be sent/received, p_x is the payload per packet and HEADER the size of the header of each packet, whose size depends on the network protocol.

I/O modeling
Modellng of energy consumption of I/O components is very important for data-intensive applications that needs to constantly read and write a big amount of data from/to I/O devices. In this work, we focus on I/O from disk. For power draw of disk operations, we use the model proposed by [225, 207]

$$P_{\text{io}}(h,t) \quad = \quad \gamma(h) load_{\text{io}}(h,t) + \mathscr{C}_{io}(h), \tag{5.18}$$

where $\gamma(h)$ models the linear relationship between disk load and power draw coefficients. Concerning $load_{\text{io}}(h,t)$, it is modeled as

$$load_{io}(h,t) = \frac{BW_{io}(h,t)}{BW^{max}_{io}(h)}, \tag{5.19}$$

where $BW^{max}_{io}(h)$ is the maximum storage bandwidth on the PM h and $BW_{io}(h,t)$ is the bandwidth at the time instance t.

Thermal modeling

Modeling the thermal behavior of data centers is an essential first step to the design of effective thermal management techniques. In [226], authors present a *spatio-temporal* analytical model to characterize the thermal behavior of datacenters, thus allowing the resource management system to make fast and online scheduling decisions in real time. The spatial model describes the correlated behavior for the inlet temperatures of all the servers in the room, while the temporal model describes the temperature evolution for an individual computing node inside each server. Traditionally, these two different types of temperatures have not been considered simultaneously by the literature: the inlet temperature is often linked to cooling optimization, while the node temperature often comes as an optimization objective or constraint. Energy consumption of cooling systems mostly depends on the technology used, as described by [227]. In our work, we focus only on air-cooled Ultrascale systems, as in [228]. Power consumption of cooling on a host h depends on two main quantities: the heat generation, that is proportional to CPU utilization, and the consumption generated by fans. To model its power consumption, we slightly modify the model provided by [229] as described in Equation 5.20

$$P_{cool}(h,t) = \zeta load_{cpu}(h,t) + c_2 T_{die}(t) + c_3 T_{die}^2(t) + P_{fan}(h,t) + \mathscr{C}_{cool}(h) \quad (5.20)$$

where $load_{cpu}(h,t)$ is modeled as in Equation 5.12. $T_{die}(t)$ and $T_{die}^2(t)$ represent the temperature of the die at time instant t, $P_{fan}(h,t)$ the power consumption of the fans at instant t and $\mathscr{C}_{cool}(h)$ a constant related to idle consumption of cooling system. Regarding $P_{fan}(h,t)$, it is modeled as in [229]

$$P_{fan}(h,t) = 0.0012 RPM(h,t) - 12 \times 10^{-8} RPM(h,t)^2 + 28 \times 10^{-2} RPM(h,t)^3 \quad (5.21)$$

where $RPM(h,t)$ are the revolutions per minute of the fans on host h at time t.

5.1.5 Runtime modeling

Precise models of the runtime conditions are needed for performance evaluation purpose. These models encompass the whole utilization and context of ultrascale infrastructures: Workload and Resource management systems.

Workload modeling

The workload modeling consists in providing the laws and characteristics of possible tasks submitted in Exascale infrastructure.

As described in [230], most known workloads are quite small as they are acquired on currently existing infrastructure, meaning only several Petaflops of computing power. Due to the envisioned large number of processing elements in Exascale computers, it is necessary to use alternate sources of data in order to propose adaptation to existing laws.

As an example, in [231], authors use traces from a large computing cluster of Google in order to evaluate tendencies on workload laws. One of the main teaching

comes from the large number of failing tasks which is aligned with the expected high failure rate in multi-million core systems.

Overall the classical laws for workload modeling are: task dynamism (the inter-arrival time of tasks) and for each category: Frequency (the sum of all frequencies of a particular workload must be 1); Mass which represent the mean amount of computation ie the makespan; Disparity which is the ratio between the mean and median makespan, and thus must be greater than 1.

Usually, the models used in the literature [230, 231] are:

Task dynamism Inter-arrival time follows an exponential law
Makespan follows a lognormal law taking into account the mass and disparity
Frequency follows usually a normalized powerlaw between all the classes

One of the key missing element of the classical approach is the missing link between the quality of service of past tasks and the submission of future ones. In [232], authors demonstrate that depending on the efficiency of past decisions of the scheduling, authors will submit more or less tasks. They propose to have a feedback between the quality of service of past tasks and the submission rate of present tasks.

Model evaluation

In this section we perform a preliminary validation of the selected model. First, we compare our model with other state-of-the-art models, by measuring energy consumption during the execution of benchmarks described in [225]. For measurements, we employ Voltech PM1000+ power measurer. Concerning non-live traces, we show the consecutive execution of CPULOAD-SOURCE and CPULOAD-TARGET benchmarks. Concerning live migration, we show the consecutive execution of CPULOAD-SOURCE, MEMLOAD-VM, CPULOAD-TARGET, MEMLOAD-SOURCE, and MEMLOAD-TARGET. The comparison is performed by implementing different models inside DISSECT-CF simulator and comparing their results with our model. For this reason, we consider only CPU, network and I/O coefficients, since data about memory and thermal are not available in the simulator. Validation results shown in Table 5.1 are obtained using a test set of 80% of the measurements and show a NRMSE below 7% for both Opteron and Xeon data set, with an average MAE of 18.28W on the Opteron machines, and of 11.5W on the Xeon machines. First, we compare the piecewise linear prediction with the linear model in [233] showing a reduction in NRMSE of 9.4% (15.6% versus 6.2%) on Opteron machines and of 7.1% (13.9% versus 6.8%) on Xeon machines. The improvement is even higher compared with the cubic model in [234], with a reduction in NRMSE of 17.4% (23.6% versus 6.2%) on the Opteron machines and of 13.6% (20.4% versus 6.8%) on Xeon. Secondly, we see in detail the results obtained by our model with the real energy traces. The MAE and NRMSE error metrics is computed on both instantaneous power and energy consumption on both sets of machines. We also compare in Figure 5.2 (for the Opteron set) and Figure 5.3 (for the Xeon set) the power traces obtained from the simulator with the real measurements.

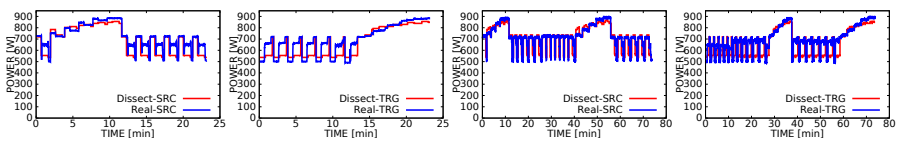
We observe that the simulation is able to provide power values with a MAE not higher than 67.3W compared to the real ones. This value is, however, influenced by the fact that in some cases the power consumption is underestimated by around 100W

Model	*Machine set*	$\mathscr{L}$	α	β	γ	δ	P_{idle} $[W]$	$P_{\max}$ $[W]$	*MAE* $[W]$	*NRMSE* [%]
Our Model	Opteron	0.12	5.29	0.68	0.05	0.1	501	840	18.28	6.2
	Xeon	0.12	4.33	0.47	0.05	0.1	164	382	11.5	6.8
Linear	Opteron		284.974	618.67					47.8	15.6
	Xeon		165.08	212.563					23.7	13.9
Cubic	Opteron		209.317	695.684					69.2	23.6
	Xeon		140.753	235.776					37.3	20.4

Table 5.1 Model coefficients and errors.

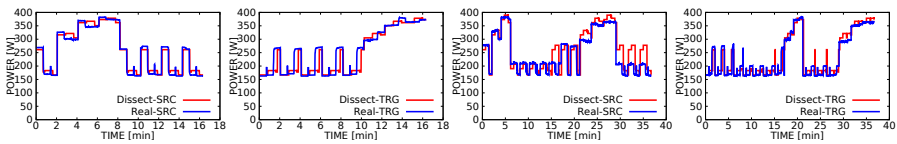
like in Figure 5.2a (between 12 and 24 minutes and Figure 5.2b (between 0 and 14 minutes), because the test scenarios active during those minutes perform non-live migrations while both hosts are idle.

In these situations, the simulator only considers the power consumption caused by the network and storage despite some inherent CPU consumption caused by these two operations. Thus, the simulator considers idle CPU consumption for both hosts, despite slight CPU load caused by the need for supporting the storage operations. This slight load, on the other hand, leads to significant offsets in the power consumption model according to the Figure 5.1. Nevertheless, NRMSE is in each case between 8% and 22% for instantaneous power consumption, and between 8% and 25% for energy consumption, showing that our model is able to provide good accuracy, that will be even higher by including data from other subsystems.



(a) Non-live migration source. (b) Non-live migration target. (c) Live migration source. (d) Live migration target.

Figure 5.2: Results for the Opteron machine set.



(a) Non-live migration source. (b) Non-live migration target. (c) Live migration source. (d) Live migration target.

Figure 5.3: Results for the Xeon machine set.

5.1.6 Simulation

In order to be able to evaluate the quality of decision making systems, it is also important to have a clear framework of analysis. Due to the scale of Exascale systems, it is difficult to evaluate on actual infrastructures. Simulation is complex at these sizes and thus a large variety of techniques have been developed. Several simulation framework propose models for resource management. Simgrid[235] and CloudSim[236] proposes to simulate tasks and the infrastructure they run on from the point of view of raw computing power. GroudSim [237] propose a simulation engine for Grid and Cloud, but energy of thermal management is not taken into account. On the Cloud federation side, DISSECT-CF [238] proposes a model for energy consumption considering different components of PMs, without however considering RAM and thermal management. Some other simulators such as DCWorms[239] also propose to take into account during the simulation the thermal effects.

Current simulation frameworks have mainly two limitations:

Scale Most simulators simulate independently each computing element such as the cores leading to consuming too much time to simulate Ultrascale systems. Stochastic simulations[240, 241]

Coverage No simulator is able to cover all the needed elements for Ultrascale simulations. Some works are trying to add the missing elements such as power and energy[242], Dynamic voltage and frequency scaling (DVFS)[243] or thermal simulation[239]. These improvements are in different simulators leading to difficulties to obtain a precise and global simulation taking into account all phenomenon.

5.1.7 Issues and open challenges

In this section we identify the issues and open challenges in energy modeling and simulations for Ultrascale systems. The main issues with energy modeling relies in the fact that while a lot of literature covers modeling for energy consumption of CPU, very few work covers with the same accuracy the modeling for other components of PMs, such as RAM, network, I/O and thermal. The main reason behind this is that until latest years, computation, and therefore CPU has been the most energy impacting component. However, with the advent of big data and IoT, always more data are transferred through the Internet to be processed and extract meaningful knowledge out of them. For this reason, it is reasonable to believe that transfer of data over the network, and consequently I/O and RAM, will have an increasing importance for energy consumption. Therefore, there is a strong need of precise models also for these components, to be able to model energy consumption of applications handling big quantities of data. Also, thermal modeling is often neglected in Ultrascale systems modeling, regardless of its impact on energy consumption. Such modeling would enable the use of effective thermal management systems, that would allow to increase energy efficiency. Moreover, at the moment there are different models for energy consumption of Ultrascale systems, with a consequent lack of a unified reference model about their energy consumption. As a consequence, each system employs different energy models, making difficult to identify energy consumption of Ultrascale

systems. For this reason, it is important for the Ultrascale system research community to develop a reference model for energy consumption of each one of these components and therefore for energy consumption of complete Ultrascale systems.

This lack of a unified view is reflected also by the current state-of-the-art simulators for Ultrascale systems: at the current state, existing simulator either focus on specific sources of energy consumption or on very specific scenarios. Also, there are several scalability issues for this type of systems, targeting both event-based and stochastic simulations. For this reason, research on simulators for Ultrascale systems should provide a unified simulation framework for this type of systems. Such unified framework should be able to simulate the behavior of all the sources of energy consumption in these systems, providing an accurate simulation that will help both scientists and application designers in identifying energy consumption of these systems.

5.1.8 Conclusion

In this work, we have identified the state of the art in terms of energy modeling and simulations for Ultrascale systems. First, we describe a model for a Ultrascale system, with special focus on Cloud data centers, then we move to modeling of individual PMs and switches. Once identified the main components to be modeled and the main challenges in their modeling, we describe how perform simulations of energy consumption in Ultrascale systems. Finally, we identify open challenges in energy modeling and simulation. In the future, we aim at proposing a holistic view of energy modeling, in order to capture the energy consumption of all components of Ultrascale systems. This way, we aim at have a fine-grained precise modeling of the whole system, in order to identify sources for energy leakages and pro-actively take actions to reduce them, thus increasing Ultrascale systems' energy efficiency. Concerning simulations, we aim at bring such holistic view of modeling in the design of Ultrascale simulations. In this way, it will be possible to identify energy consumption in Ultrascale systems and applications before they are running and improve them before the time they are deployed.

Acknowledgements

The work described in this book chapter has been funded through the Haley project (Holistic Energy Efficient Hybrid Clouds) as part of the TU Vienna Distinguished Young Scientist Award 2011 and Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015.

5.2 Evaluation of Renewable Energy Usage for an Exascale Computer

Supplying datacenters with clean-to-use renewable energy is essential to help mitigate climate change. With renewable energies in datacenters, Exascale can then be achieved without (or with less) greenhouse gases emissions. Renewable energies

include solar panels, wind turbines ; most of them are intermittent and difficult to predict in the long-term. Storage elements like batteries, fuel cells and supercapacitors are then necessary. These storage units are complementary. Supercapacitors, with their low energy but high power, handle very short term fluctuations of power and ensure the stability of the system. Batteries, with a much higher energy capacity, enable shifting load or generation through time, but only over a few minutes or hours. Finally, to account for longer term variations and seasonal trends, hydrogen storage (combining an electrolyzer to generate hydrogen from electricity, hydrogen storage tanks, and fuel cells to generate electricity from hydrogen) is also used. Many works have been made on power sources modeling: the models proposed define the characteristics and behavior of the components. These models must be sufficiently representative of reality while remaining easily usable. Different models are possible depending on the use and the needed granularity. For example, an aged fuel cell only delivers a fraction of the output of a newer one. To account for this phenomenon, aging can be considered in the proposed models. These models are then used in works concerning tasks placement and energy saving; depending on the models used, decisions can include capital (replacement) costs, and not only operation costs [244]. Lot of researches are currently done on optimization of the IT load under renewable energy constraints. It aims to manage the workload in such a way that power consumption matches as closely as possible to the power available through renewable energies. For example, batch jobs may be delayed, in order to run when more energy will be available. Most of the approaches propose a centralized optimization with a global knowledge from the IT load and electrical point of view. Contrary to this, the approach presented in [245] considers the power supply as a black box which only provides an energy budget to the IT side where the IT load is optimized under renewable energy constraints. In [246] authors propose to separate the optimizations of electrical infrastructure from the optimizations of the computing resources. They designed an online greedy scheduling algorithm (called Attractiveness-Based Blind Scheduling Heuristic, or ABBSH), which exchanges information with the electrical management system in order to take availability of energy in consideration. Along with this algorithm, several multi-objective functions are proposed to handle the trade-off between energy and performance considerations.

More and more data centers are built in cold climate areas for cheaper cooling and increased energy efficiency of the facilities. However, such geographical locations have highly varying availability of renewable energy (especially solar energy), and fitting the data centers completely with renewable energy is challenging. In this subsection we analyze the feasibility of using renewable energies for a data center located on 60°north latitude and discuss the competitiveness of deploying green datacenters in different geographical locations. We also provide an analysis on the impact of battery size on the green coverage percentage, green energy loss, and brown energy taken from the traditional grid. Figure 5.4 represents an overview of the energy sources included in the following paragraphs.

The remaining of this chapter presents some examples of renewable energy modeling and an algorithm to schedule tasks in this context.

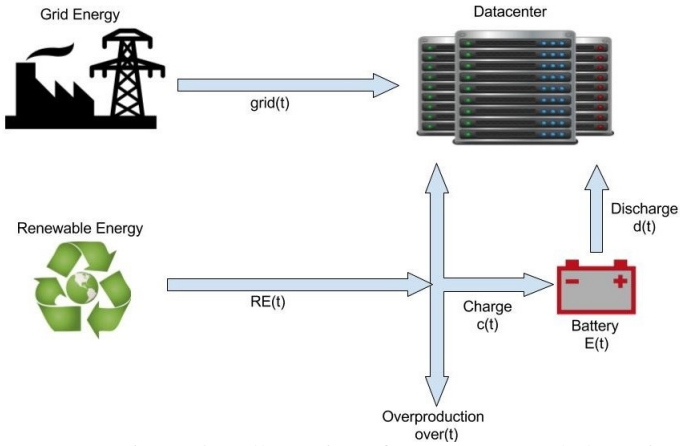


Figure 5.4: Illustration of energy sources balance in our study.

5.2.1 Renewable energy modeling

This subsection outlines the total amount of renewable energy produced in one year for a 60°geographical location: Turku, Finland. In order to calculate the amount of renewable energy we consider only renewable energy produced by wind turbines and solar panels. Since the weather and the season directly influences the production of renewable energy, we must utilize a weather model to predict the production rate. The weather data are collected from a weather station located at Åbo Akademi University in Turku, Finland. To describe the production rate of a solar panel, we acquire the data containing the solar power radiance on a horizontal 1 m^2 solar panel, and we calculated the produced power in Watts. We also acquire the data regarding the wind speed in $[m/s]$, and model the resulting power production for a wind turbine.

5.2.1.1 Solar power model

.

The power incident on a solar panel depends not only on the power contained in the sunlight, but also on the angle between the module and the sun. Referring to [247] we calculate that for a 60°of northern latitude, representing the location of Turku, the angle at which a solar array should be tilted in order to achieve maximum energy through all the year is 45°with respect to the horizontal plane. We assume that the solar array tracks the sun on the vertical axis (east to west), but is fixed on the horizontal axis. Equation 5.22 shows the power generation of a 1 m^2 solar panel as:

$$P_{\text{solar}} = e \cdot ie \cdot P_{\text{solar\_h}} \cdot \frac{sin(\alpha+\beta)}{sin(\alpha)} \tag{5.22}$$

where $P_{\text{solar\_h}}$ is the solar radiance in the horizontal plane we already have from weather data, α is the sun elevation angle through the year and β is the tilt angle

of the module measured from horizontal plane, 45°. The value for α is calculated according to Equation 5.23:

$$\alpha = 90 - \phi + \delta \tag{5.23}$$

where ϕ is the latitude (60°) and δ is the declination angle computed in Equation 5.24 as:

$$\delta = 23.45^\circ \cdot sin[360 \cdot \frac{(284+d)}{365}] \tag{5.24}$$

where d is the day of the year. In addition, e represents the solar panel efficiency which is the percentage of the sunlight energy that is actually transformed into electricity because of limitations in the solar panel cells. In order to achieve realistic data, and according to the latest research, a value of 0.18 is chosen to multiply all hourly solar energy values. The Solar inverter efficiency *ie* represents the efficiency of the inverter connected between the solar panel cells and the AC grid. The average coefficient of the DC-AC power converting today is 95%. This value is taken into account to assure accurate and realistic power values.

5.2.1.2 Wind power model

The wind power model describes the power generation from the wind turbines in the system. To produce the wind energy we have chosen a HY 1000, 5 blade wind turbine generating a peak output power of 1200 W. The wind power model is constructed by taking into consideration the following key features:
Wind turbine power curve: According to the power profile in the technical specifications, we constructed the mathematical model of power as a function of wind speed. Equation 5.25 describes the power production of a wind turbine as follows:

$$P_{\text{wind}} = ie \cdot 1151 \cdot exp[-[\frac{(speed_{\text{wind}} - 14.28)}{6.103}]^2] \tag{5.25}$$

where $speed_{\text{wind}}$ is the wind speed in $[m/s]$ and *ie* represents the solar inverter efficiency, typically reaching a value of 95%. The parameters in Equation 5.25 were obtained by using curve fitting tools in Matlab and using the power data from [21] as a reference. Having this formula, we obtain the annual hourly wind energy.

Finally, the total renewable power model is given as the sum of the total solar and total wind production:

$$P_{\text{renewable}} = P_{\text{solar}} + P_{\text{wind}} \tag{5.26}$$

The overall annual renewable energy is calculated to be 1358.27 kWh, 402.2 kWh generated by 1 m^2 solar panel and 956.16 kWh by one HY 1000 wind turbine respectively.

5.2.1.3 Renewable quantity model

The energy consumption, which represents the total energy consumed by the servers and the cooling system, is evaluated through simulation and equals to 253 MWh per year. The data center is composed by 1000 physical machines and 2000 virtual machines running a synthetic workload. Detailed information about the used model

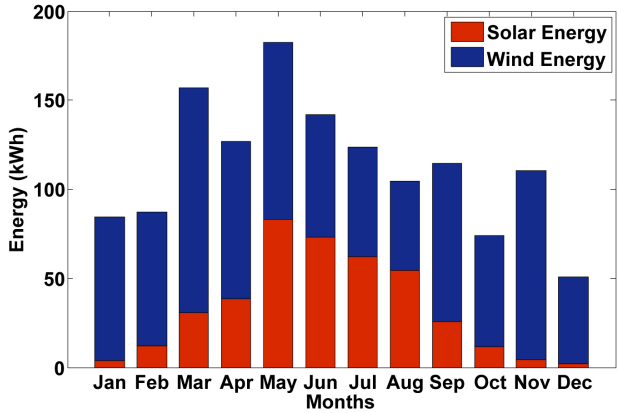


Figure 5.5: Monthly available renewable energy over a year produced by 1 m^2 solar panel and 1 HY 1000 wind turbine

is available from [248]. We build the model which describes the relation between the number of wind turbines and square meters of solar panels required to ensure a desired value of coverage with renewable energy. Figure 5.6 shows the relation between the number of turbines and number of m^2 solar panel needed for green coverage of 25%, 50%, 75%, and 100%, over one full year.

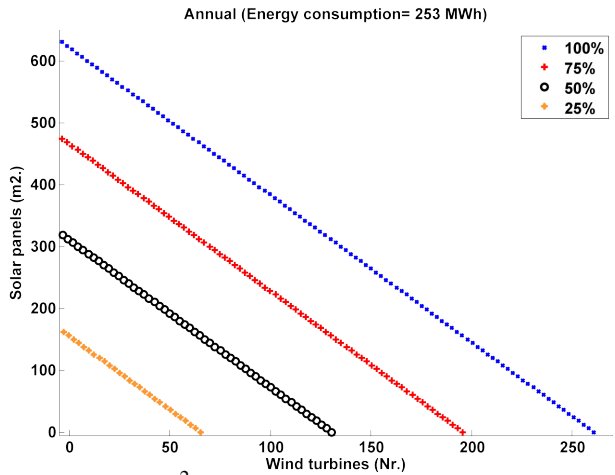


Figure 5.6: The number of m^2 solar panel over the number of turbines for 4 green coverage levels over one year

5.2.1.4 Minimum Percentage Supply

A new metric is introduced in order to measure the ratio between the renewable energy being produced by 1 turbine and a 1 m^2 solar panel and the data center energy consumption. It is called *Minimum Percentage Supply* (MPS), and it is calculated as:

$$MPS = \frac{RenewableEnergyProduction(kWh)}{TotalEnergyConsumption(kWh)} \cdot 100 \tag{5.27}$$

For five different experimental scenarios, with physical machines varying from 500 to 2500, we calculate the average value of MPS over a year, and for specific months in Finland when renewable energy hits the maximum (May) and the minimum (December). As energy consumption increases, MPS decreases nearly linearly. Table 5.2 presents the annual, minimum and maximum month MPS values, for 5 different datacenter size. As seen in Table 5.2, there is a 3 fold difference between minimal and maximal MPS values, which clearly indicates different operational costs for producing the same amount of renewable energy during different times of the year. Obviously, more physical resources, i.e wind turbines and solar panels, are needed in December to produce same amount of energy compared to May.

Table 5.2 MPS annual, maximal and minimal months values in percentage

Nr. of physical machines	500	1000	1500	2000	2500
Annual MPS(%)	1.10	0.54	0.36	0.27	0.21
May MPS(%)	1.70	0.85	0.57	0.42	0.34
December MPS(%)	0.47	0.24	0.16	0.12	0.09

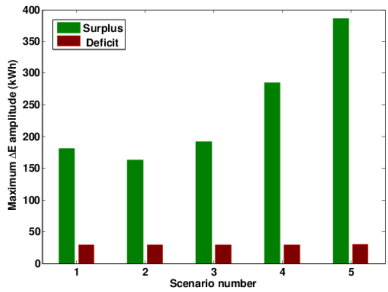
5.2.1.5 Cost evaluation

The renewable energy production fluctuates between different time intervals depending on the weather conditions. The data center consumption fluctuates between different time intervals depending on the workload as well. This means that occasionally there is a higher production rate than consumption rate and occasionally it is the opposite. We refer to the difference between renewable available energy and the workload energy consumption as ΔE, given as follows:

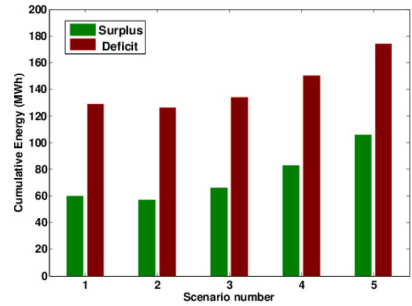
$$\Delta E = \textit{Renewable energy - Energy consumption} \tag{5.28}$$

We refer to $\Delta E > 0$ as *surplus*, (or overproduction) meaning that more renewable energy is produced than what is consumed. This additional energy can, for example, be stored or be sold back to the grid if allowed. And, we refer to $\Delta E < 0$ as *deficit*, meaning that consumption of energy is higher than the production and, consequently, external *brown* energy must be bought from the grid.

Referring to periods of time during a year when renewable exceeds the consumption as surplus hours, the results of the study show that combining wind and solar energy sources provides greater surplus hours compared to using only one source of renewable energy. Meanwhile, increasing the number of solar panels and decreasing



(a) Amplitude of ΔE over one year.



(b) Amplitude and cumulative energy over one year.

the number of wind turbines, leads to decreasing the surplus hours in a year. However, during these shorter surplus periods, the excessive energy surpasses the value of energy consumption by 6-13 times. This brought us to two logical interpretations. The excessive energy represents wasted energy in case it won't be reused or income in case it will be stored in batteries and/or sold back to the grid.

In this perspective, we further investigate the cost aspect of using renewable energy, calculating overall income and expenses of combined grid and renewable energy in a data center, over one year. We show that using only solar energy source leads to extreme energy and cost values. This results in high income during short surplus periods of the year. Also, when solar energy is lacking, expenses are high in order to provide for the data center energy need. The results also show that mixing wind energy source with a certain portion of solar, to compensate the lack of wind, is the opposite. Therefore, there is less, but more stable income.

Figures 5.7a and 5.7b present respectively the achieved ΔE amplitudes and annual cumulative energy surplus and deficits for the scenarios defined in Table 5.3. Values are based on a data center composed by 1000 physical machines and 2000 virtual machines running a synthetic workload. Detailed information about the used simulation environment is available from [248].

Table 5.3 Number of wind turbines and solar panels for 5 selected case study scenarios

Scenario Nr.	1	2	3	4	5
Wind Turbines Nr.	195	150	100	50	0
Solar Panels Nr.	0	108	228	348	465

5.2.2 Battery modeling

The production of energy from renewable resources, such as sunlight and wind, is variable over time and dependent of weather conditions. One common way to address

this problem is to use batteries, not only as a backup in case of energy outage or as a power peak shaving, but as an energy storage device. Latest research proposes the usage of batteries as a key source of energy for the energy system supply of a datacenter. In this case, the main question to be tackled is: how to choose the battery capacity to maximize the renewable energy usage thus minimize the green energy loss. The concept of green coverage mentioned also in subsection 5.2.1.3 represents the percentage of total energy consumption provided by renewable energy.

We consider a datacenter consuming energy, which is provided by one of these three sources: renewable energy, battery energy and grid energy, according to this order of priority. The battery gets charged only by the renewable source and it charges only when the renewable energy is of greater amount than the datacenter energy needs. If the battery is full and there is still renewable energy produced and therefore not being used, this extra energy is considered overproduction. The battery discharges when datacenter energy consumption is higher than what is provided by renewable sources. In cases when both, renewable sources and the battery, are not enough to fulfill the datacenter energy requirements, additional energy is taken from the traditional grid (also referred to as brown energy).

5.2.2.1 Simulator tool

In order to analyze and being able to predict the amount of available energy in the battery over specific moments in a chosen time period, we developed a simulation tool. The simulator is based on the concept of the battery as a finite state machine, whose trigger conditions are related to the available amount of solar energy and datacenter energy consumption at a specific time t.

There are four possible states that the battery can be at any time t: Full, Discharging, Charging or Empty. Therefore, there are 16 possible state transition combinations. Practically, all of these combinations are feasible except for the Full - Empty and Empty - Full transitions which are generally limited by the charge/discharge rate of the battery during a certain period of time. Possible triggers from one state to the other depend on the amount of available solar energy in a moment of time t, referred to as *RE(t)* (Renewable Energy) and the datacenter energy needs in that moment t, referred to as a *consum(t)*. Other affecting factors are the value of the stored energy in the battery, *E(t)* and the maximum energy capacity of the battery named *Efull*.

Implementation: The pseudocode for developing the simulator is given below as a set of 8 steps. *BS(0)* refers to the initial Battery State assigned to Full, assuming the battery is fully charged when the simulation begins running. Each of the 16 'current - next' state combinations is assigned a combination number (named *combinationNr*), which calculates the energy (*E(t)*) the battery will have on every t. The green coverage is calculated according to Equation 5.30 and printed out. The loop repeats 8760 times for every hour of the year, resulting in outputs of total charging and discharging amount of the battery, overproduction, and grid energy for every hour.

The equation 5.29 is checked by the simulator to be true for every moment of time t during the simulation time period.

$$RE(t) + d(t) + grid(t) = consum(t) + c(t) + over(t) \tag{5.29}$$

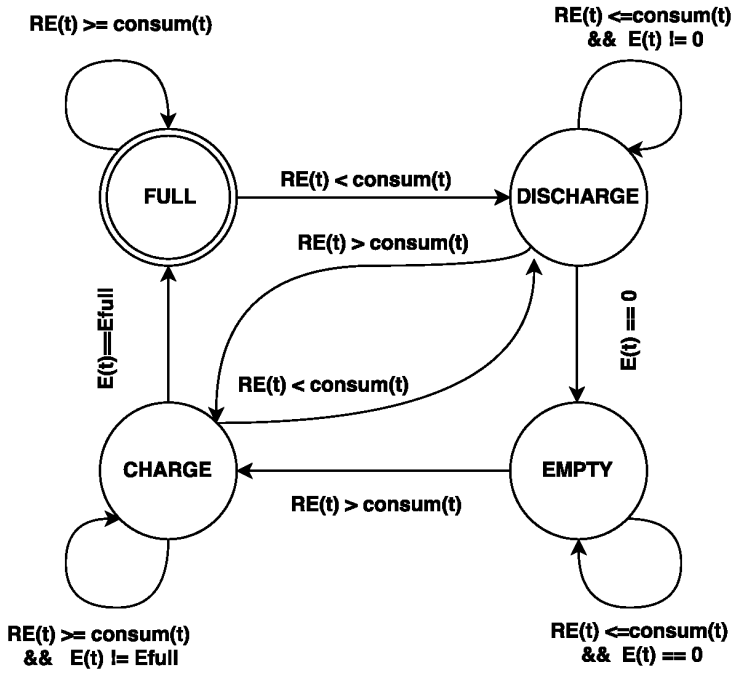


Figure 5.7: Battery states over time represented as a finite state machine.

Algorithm 3 Simulator tasks

1: $BS(0) \leftarrow FULL$, $t \leftarrow 1$
2: **repeat**
3: Define $BS(t) = f[BS(t-1), RE(t), consum(t)]$
4: Define $combinationNr = f[BS(t-1), BS(t)]$
5: Calculate $E(t) = f[E(t-1), combinationNr]$
6: **until** $t \leq 8760$
7: $greenCoverage$ = annual $[RE(t) - over(t)]$/annual $[consum(t)]$
8: Print $charge(t), discharge(t), overproduction(t), grid(t), greenCoverage$

On the left side of Equation (1) are listed the providing energy sources and on the right side is listed drawn energy. $RE(t)$ represents the renewable energy being produced at moment t, $d(t)$ is the amount of energy discharged from the battery in the time period $(t-1)$ to t, and $grid(t)$ represents the amount of energy taken from the traditional grid at that moment t, in case it is needed to fulfill the datacenter energy needs. The variable $consum(t)$ refers to the datacenter energy consumption at moment t, $c(t)$ is the amount of energy charged to the battery in the time period $(t-1)$ to t and $over(t)$ represents the part of the produced renewable energy which is not consumed neither by the datacenter nor from used to charge the battery. Based on Equation 5.29 we calculate the green coverage metric of our specified datacenter over a year, as described in Equation 5.30. The sum is composed of 8760 hourly values for

each of the metrics: $RE(t)$, $over(t)$ and $consum(t)$. We can distinguish two different scenarios from this equation. First, theoretically we evaluate the green coverage simply as total renewable energy over the year divided by total energy consumption over the year. We have no information regarding the overproduction without running the simulations, so we assume it is zero. Second, experimental simulations show that the overproduction is greater than zero, meaning that the real green coverage is less than the theoretical calculated value. The battery size is the key element affecting the minimization of the overproduction value.

$$greenCoverage = \sum_{t=1}^{8760} [RE(t) - over(t)] / \sum_{t=1}^{8760} consum(t) \tag{5.30}$$

5.2.3 Geographical location competitiveness

In this subsection we investigate how different geographical locations affect the green coverage and the required battery size to reach a certain green coverage. Given that only solar energy is affected by the latitude factor, we will consider as renewable energy only the solar energy produced. To highlight the differences to northern latitudes we choose Crete, Greece at 35°latitude because of its typically solar intense southern European climate, and Llorin, Nigeria at 8.5°latitude to cover the equatorial extreme point. Details on the renewable energy production for each of the countries can be found in [249].

5.2.3.1 Solar energy availability

The total amount of renewable energy provided over a year by each of the three selected countries: Finland, Crete, and Nigeria, is given in Figure 5.8. Table 5.4 presents numerically the total amount of annual solar energy provided by a 1 m^2 solar panel and the required quantity of solar panels in m^2 for the three countries, aiming for half of the datacenter energy consumption to be provided by the solar energy source. On closer observance of the data, we notice a fair similarity between Nigeria and Crete, with only a 17% difference. Almost twice (45%) of the number of solar panels needed in Nigeria are needed in Finland to achieve same target of solar energy production.

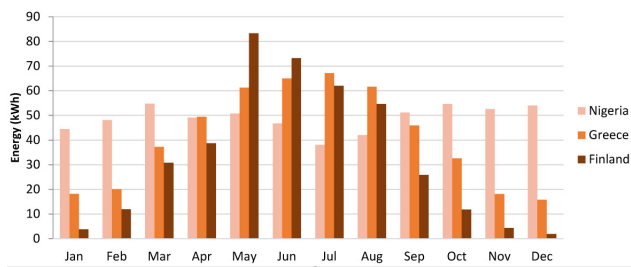


Figure 5.8: Solar energy produced by 1 m^2 solar panel in three geographical locations monthly over a year

Table 5.4 The total amount of annual solar energy provided by 1 m^2 solar panel and the number of solar panels in m^2 needed to cover 50% of energy consumption for Finland, Crete and Nigeria.

Country	Annual Solar Energy/1m^2 (kWh)	Required solar panels (m^2)
Finland	402	324
Crete	500	260
Nigeria	585	222

5.2.3.2 Battery size for 3 different locations

Experiments and Results To perform the experiments we have run the simulation tool by changing the input of battery size from 0 to 10MWh, separately for each of the countries. The first run under battery size equal to 0 shows how much from the produced solar energy is spent in vain as overproduction. Increasing the size of the battery means increased amount of stored solar energy. The yield of this is diminishment of the amount of overproduction and energy taken from the grid, translating to an overall higher renewable energy usage. The optimization concept based on the battery size is related to the fact that as battery size increases, the amount of energy taken from the grid and the overproduced energy is decreased. The decreasing rate depends on the battery capacity and of the energy production pattern, which differs in the selected geographical locations. The simulation time covers a period of 1 year (8760 hours), taking as input the hourly solar energy records and the hourly energy consumption values over the year. The datacenter is composed of 100 servers and we assume in this paper a synthetic variable workload over one week repeated over the whole year. The annual energy consumption for this datacenter equals to 260 MWh. The used synthetic variable workload is obtained as presented in [250].

Through empirical simulations we found that to achieve no solar overproduction over a year, a 230 kWh battery capacity is needed for the datacenter located in Nigeria, a 292 kWh battery capacity for the Greece location, and a 9.3 MWh battery capacity if location is Finland. According to a typical Tesla Powerwall 2 battery chosen as a template with a capacity of 13.5 kWh [251], the overall battery size needed is equal to 17, 21, and 688 of such batteries for Nigeria, Crete, and Finland respectively. According to Equation 5.30, the real green coverage is decreased from the theoretical one in proportion to the overproduction amount. For a theoretical green coverage for the 3 countries, we chose to assign the same value of 50%. The first simulation with battery size 0 for each of the countries shows lower values of green coverage because of the overproduction energy, illustrated in Figure 5.9. The achieved values are 42%, 35%, and 31%, for Nigeria, Crete, and Finland respectively. This means that out of a desired annual renewable energy usage of 50%, there is an 8%, 15%, and 19% drop from this desired usage because of the total overproduction energy over the whole year. The way to recover from this problem is increasing battery size. Figure 5.8 graphically shows higher values of green coverage rising towards 0.5 (50%) with

increasing battery size. The rate by which this increase happens changes by country, referring to the annual values.

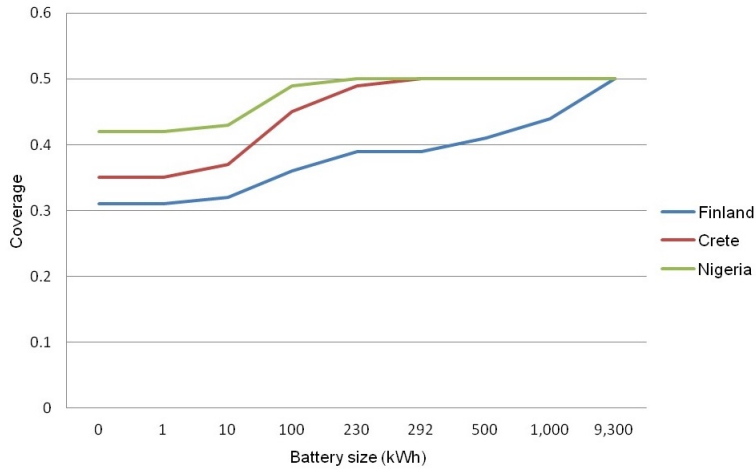


Figure 5.9: The value of achieved average green coverage over a year with increasing battery size for Finland, Crete, and Nigeria.

5.2.4 Renewable energy and Exascale computing

One of the constraint limiting the development of Exascale computing is the power consumption. To achieve an Exaflop computer a limit of 20MW was set on its maximal power dissipation by the U.S Department of Energy (DOE) to stay behind cost feasibility limits. When looking at the current power efficiency of supercomputers from the Top500 [46] and Green500 [252] lists, it clearly illustrates how ambitious is this challenge. On average the 5 most powerful computers from the Top500 list achieve around 36 teraflops for a power dissipation of 5.6MW. Scaling these ch5/fig-sec5-3 to an Exaflop, taking into account the 20MW limit, would require 7 times more power efficiency computing technologies. On the other hand, when taking as reference the average efficiency of the 5 most efficient computers from the Green500 list, it would require 3.5 times more power efficiency computing technologies to provide an Exaflop with a power budget of 20MW.

Even if the 20MW limit might sounds a challenging goal, Koomey and al. established that electrical efficiency of computation has doubled roughly every year and a half for more than six decades [253]. This observed trend indicates that it could be possible to provide an Exascale computer dissipating 20MW from 3 to 6 years from now.

When scaling the results from [248] we can evaluate the required number of solar panels and wind turbines to provide different coverage of renewable energy sources for an Exascale computer dissipating a maximum power of 20MW and deployed in Turku, Finland. Here we assume a workload of users requests uniformly distributed over a 24 hours time window. This workload is replicated for every day of the year,

assuming an homogeneously distributed workload pattern over the year. The assumed wind turbine is a 1kW HY 1000 wind power characteristics. Detailed description about the simulation settings and assumptions are available from [248]. Table 5.5 provides the number of required solar panels and wind turbine to cover 50% of the energy consumption of the Exascale computer over a one year period. Without any wind turbine, the required surface of solar panel would be 220 000 square meters, which is equivalent to 31 football pitchs.

Table 5.5 Required wind turbines and solar panels to provide a 50% renewable energy coverage

Scenarios	1	2	3	4
Wind turbines (Nr)	0	33 300	60 000	86 500
Nr. Solar panels (m2)	220 000	133 000	66 600	0

5.2.5 Policy for renewable-powered datacenters

As seen in the previous subsections, the problematic of renewable energy is becoming a major focus for powering datacenters. Several international actions shows this importance.

5.2.5.1 European Commission

Due to the European climate and energy package called 20-20-20, there is a goal to reach the following targets:

- A 20% reduction in EU greenhouse gas emissions from 1990 levels;
- Raising the share of EU energy consumption produced from renewable resources to 20%;
- A 20% improvement in the EU's energy efficiency.

These goals are not only for reducing ecological footprint of European activities but also with an economical goal of creating jobs.

In this context, several funding programs were created in order to improve the overall energy efficiency of European Community countries with one on the focus of large scale datacenters : *Datacenters in Smart Cities* funded in 2013. This funding schema came after the success of several independent EU-funded projects for energy-efficient datacenters such as Fit4Green[254], Games[255], All4Greens[256], CoolEmAll[257] and Eco2Clouds[258].

These European projects focus on the interaction of datacenters with their electrical surroundings such as battery, solar panels, ...

5.2.5.2 U.S.A.

One of the main driver of the USA policy comes from the National Strategic Computing Initiative (NSCI) which assigned the U.S. Department of Energy (DOE) Office of

Science (SC) and the National Nuclear Security Administration (NNSA) to create a framework to build an Exascale supercomputer.

Several funding scheme came from this objective but the main focus was put on energy efficiency and energy- and power-capping, not on using renewable sources.

5.2.5.3 Collaborative projects

A large number of international or national projects have the focus to evaluate the possibility of powering datacenters with renewable energy.

DataZero [259] French funded project aims at powering a large scale datacenter using only renewable energies without an access to an electricity grid. It focuses on the task scheduling and electrical infrastructure reconfiguration.

CtrlGreen [260] This French project addressed the autonomic management of a datacenter partly operated by renewable energy. It focused on the scheduling side of the problem.

GreenWare [261] is a NSF funded project taking into account the cooperation of a federation of datacenter to optimize renewable energy consumption.

5.2.5.4 Companies

Several large companies (Apple, Google, ...) announced using renewable energy equivalent to the energy consumed by their datacenters. At the moment each of their datacenters are not directly consuming this renewable energy but their owner are funding this production for other usages. Thus the described interconnection between renewable electrical sources and datacenters are not yet actually used.

5.3 An Introduction to Cooling in Data Centers

Data centers can require up to several megawatts of power, which are dissipated as heat. Dissipating this heat requires careful planning to minimize costs. Converged high performance and big data computing infrastructures [262] typically also require cooling. This can be a significant part of the fixed and operational costs, and typically needs to be customized to the climatic and infrastructure conditions available at a data center site. There is at present no standard methodology to do this, although there is a growing body of knowledge in this area. It has been recognized as an area of importance for the European Union with awards for the best European data centers being given out annually [263]. There are also standardization efforts underway in other parts of the world [264, 265, 266, 267].

The motivation for improved knowledge of cooling strategies is the higher density of compute units, which can make traditional fan driven air cooling a suboptimal choice. Unfortunately, most people with a high performance computing background do not have a large background in heat transfer and building design, though some engineers do use supercomputers to design thermofluid systems. The aim of this review is to introduce the main technologies and problems to be overcome in data center cooling.

The efficiency of a data center is often characterized by the power usage effectiveness (PUE), which is the ratio of the energy used for the data center to the ratio of the energy used for the computing equipment [268, 269, 270, 271, 272]. Use of the PUE as a data center efficiency metric has made cooling considerations important in data center operation since chillers and refrigeration have used upto 30% of system power [269, 273].

Previous work has focused on understanding the impact of data center ăloads on required cooling [274, 275]. This is therefore not reviewed in detail here. The data center load is an important design consideration, however for the mechanical cooling design, the most important initial consideration is being able to sustain maximum performance of the data center. More comprehensive information is available at [276]. This is a valuable openly accessible collection of resources that also targets small data centers. In addition, the Greengrid [277] and American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) handbooks [264] are useful resources, though less accessible to the casual reader.

5.3.1 Overview of Data Center Cooling Methodologies

In this section, there will be no distinction between data centers mostly targeted for cloud workloads and those targeted to high performance computing workloads. For the purposes of data center cooling, they are very similar.

The largest distinction is between air and liquid cooled data centers. A primary reason for using liquid cooling is that it allows for a higher computation density since liquids typically have a higher heat capacity than air. Liquid cooling however typically needs a more complicated infrastructure, thus may have higher setup costs.

The design of air cooled systems has stabilized. Typically an air conditioned room is used and the cabinets are arranged to have a design that would enable high heat transfer rate - usually racks are arranged to have hot and cold aisle between them to ensure good transport of hot air to the air cooling system.

There are several liquid cooling options, and this is an area still undergoing rapid innovation, though many data centers would like some standardization to enable lower site fixed costs with liquid cooling infrastructure being used for several generations of compute units. Some methods for liquid cooling that are currently of interest are:

1. Heat loop pipes, siphons, liquid cooled heat sinks and cold plates (for example produced by Asetek [278], Aquaris [279] and CoolIT [280])
2. Fully immersed forced convective cooling (for example as produced by Green revolution cooling [281] and investigated by Eiland et al [282] for the Open Compute project [283].)
3. Fully immersed free convective cooling (for example, the Rugged POD [284])
4. Phase change cooling (for example using low boiling point liquids produced by 3M [285, 286, 287, 288])

For each of these one can consider keeping the primary cooling fluid temperature either close to room temperature or significantly above room temperature (*warm water cooling* [289]). When the primary cooling fluid is significantly above environmental conditions, then seasonal temperature changes have negligible effects on efficiency

of conversion of heat for other uses and the generated heat can be more effectively recycled.

Important considerations in choosing a cooling solution are maintenance costs and ease of maintenance. In fully immersed cooling solutions, careful design is required to make component replacement easy. Furthermore, the cooling fluids need to be chosen to enable high heat transfer rates as well as ensuring system cleanliness.

In a liquid cooled system, choosing the coolant and cooling system components is again only half of the work. In many cases a secondary heat exchanger is used to transport the heat away from the computing system. Typically, the secondary exchanger will use water as the heat transportation medium. This allows for a reduction in cost and also easy integration into the building heating/cooling system, or for easy exhaust of the materials. Some of the heat may also be used to generate electricity using thermoelectric materials.

Surface properties are of great importance in heat transfer [290]. It can be challenging to model this, however such properties can be measured and the materials or material coatings carefully chosen to ensure a high heat transfer rate.

The design of data center cooling systems, typically involves cooperation between computer scientists, electrical engineers, mechanical engineers, civil engineers and plumbers - most of whom have very different technical skills. A few engineers will use simulation to help design data center cooling systems [291], however in many cases, standard design rules will be used [292]. ASHRAE has formulated data center building standards [264], though rapid innovation in this area and the adjustment to local conditions make these difficult to adopt in every possible situation.

For a simple liquid cooled design, once the data center power needs are known, the maximum, average and minimum power consumption can be estimated. Since almost all the power is dissipated into heat, one can then estimate the amount of heat that needs to be transported away. If the heat capacity of the transporting fluid is known, and the efficiency of heat generation and heat transport from the CPU, GPU and RAM to the fluid estimated, then an appropriate fluid flow rate can be calculated to ensure that the system remains within a reasonable operating temperature range. The effectiveness of the secondary heat exchanger can then be determined. Typically an electronic heat control system with temperature feedback would be used to regulate the cooling of the data center. This would move some of the implementation details for the cooling system from the design phase to the operational phase and also allow for adaptation of the system to the computational load at the data center.

In phase change cooling, the operating temperature of the chips needs to be estimated. A fluid with a low boiling point that is compatible with the electronic components is then chosen. The evaporated fluid transfers heat from the hot electronic surfaces to a liquid cooled condenser. The heat required to create a phase change from liquid to gas is known, if heat transfer between the hot electronic components and the cooling fluid, and between the evaporated cooling fluid and condenser is fast enough, the flow rate in the condenser can be adjusted to ensure adequate cooling of the electronic components. Condensers are typically custom made [293].

As concrete examples of existing deployments, table 5.6 details the characteristics of several data centers and their cooling technologies.

Site & System	Release Date	Cooling Technology	Power Capacity	Heat Reuse	Hot Water	PUE
UL, BioTech, Luxembourg	2011	Airflow	200kW	None	None	?
UL, CDC S02-1, Luxembourg	2016	Airflow	300kW	None	None	?
UL, CDC S02-2, Luxembourg	2016	Airflow	280kW	None	None	?
UL, CDC S02-3, Luxembourg	2019	DLC	1050kW	None	?	?
UL, CDC S02-4, Luxembourg	2019	DLC	1050kW	None	?	?
UL, CDC S02-5, Luxembourg	2016	Airflow	300kW	None	None	?
PSNC, Eagle, Poland	2016	DLC	450kW	Building heating	?	1.05
UT, Rocket & Vedur, Estonia	2014 2011	Airflow	88kW	None	None	?
JSCC RAS, Russia	2017	DLC	300kW	None	+45°C	1.025-1.06
SSCC RAS, Russia	2017	DLC	100kW	None	+45°C	1.05
Beskow PDC, Sweden	2015	Airflow, Water Cooled Doors	1000kW	Building Warming	+19°C	?
K computer, RIKEN, Japan	2012	DLC	12700kW	Electricity co-generation	+17°C	1.36
Shaheen, KAUST, Saudi Arabia	2015	Airflow, Liquid Cooled Cabinet	2800kW	No	+23°C	?
Taihulight, NSCC, China	2016	ILC,	15371kW	No	?	?
Ajax1, SKY, United Kingdom	?	Airflow	?	No	None	1.25
eigen, ENGIE, Netherlands	?	Airflow	?	No	None	1.28
Facebook, Lulea, Sweden	?	Airflow	?	No	None	1.1
DLTM Italy	2014	Airflow	5KW	No	None	?

Table 5.6 Example of cooling technologies deployed in existing HPC Data centers. DLC - direct liquid cooling, ILC - indirect liquid cooling, PSNC - Poznan Supercomputing center, UL - University of Luxembourg, UT - University of Tartu, JSCC RAS - Joint supercomputer center of the Russian academy of science, SSCC RAS - Siberian supercomputer center of the Russian academy of science, PDC - PDC Center for High Performance Computing at the KTH Royal Institute of Technology, RIKEN - , KAUST - King Abdullah University of Science and Technology, NSCC - National Supercomputing Center in Wuxi, DLTM - Distretto Ligure delle Tecnologie Marine

5.3.2 Cost Estimation

Cooling can be a significant part of a data center fixed and running costs. Liquid cooling will typically have a higher fixed infrastructure cost, but lower running costs. When setting up or upgrading a data center, the choice of cooling options is usually different at every site and needs to be designed specifically for that site. An understanding of local conditions (climatic conditions, electricity costs, possibilities for heat reuse), is of great importance. Also of importance is whether a new data center is being constructed or whether an old data center is being upgraded. Efficient upgrading of an old data center will try to reuse existing infrastructure and may be more difficult to adapt to a liquid cooled solution. For upgrading of old data centers, common options are to use indirect liquid cooling by utilizing cooling doors on enclosed cabinets which have components or to directly attach liquid cooled heat sinks to hot components such as the CPU, GPU and memory modules. These allow for easy servicing of the electronic components, does not require much retraining of data center personnel. They also allow for reuse of the airflow cooling infrastructure to cool components such as motherboards and storage, which require significantly lower air flows. Newer data centers which use liquid cooling can have a much better integration with building/environmental infrastructure to ensure better reuse of heat generated by the data center as well as initial training of the new personnel on best practices for operating and maintaining a liquid cooled infrastructure.

The total cost of ownership model is a popular methodology for choosing data center infrastructure [292, 270]. The choice of cooling infrastructure may be easier to model than the choices of CPU, GPU, interconnect, storage and other compute components since only a few key system properties are needed.

Some consideration are

- the outside temperature, in particular does it allow for free cooling
- can the heat be used in electricity generation
- can the heat be used to provide hot water
- can the heat be used for building heating

In cold climates, all the above uses are possible (and building heating is particularly attractive [294, 295, 296]), but in hot climates or in the hotter seasons, only electricity generation and hot water generation can be done. Each of these will likely have a different economic value.

As an example, for a 500kW data center, assuming an electricity cost of 0.05€ per kWh, electricity costs are about 300,000€ per year. With free cooling or exhaust to the atmosphere, no electricity costs are recovered - for a warm water system, chillers are not required. If hot water or building heating is provided, then only losses in heat conversion need be accounted for, these may be between 10 and 20% of system costs, thus operating costs are up to 60,000€ per year. If electricity is co-generated, assuming 35% efficiency ([297]), then electricity costs are 195,000€ per year. In cases where electricity costs are significantly less, for example electricity is generated on site, operational costs will be much less significant than the fixed cost of establishing the infrastructure.

An additional concern are component lifetimes and failure rates. Not all hardware components will be compatible with all liquid cooling solutions - in particular for fully immersed cooling solutions [298]. The cooling solution may also change the expected failure rates of computer components. On small systems, component failure rates will be infrequent, an extended warranty or similar insurance policy may be the best way to cover component failure. On large systems, component failure rates and component replacement should be considered part of the operational costs as it will be possible to apply statistical methods.

5.3.3 Simulation Tools

5.3.3.1 Electrical Load Simulation and Prediction

Due to its significant importance, cooling optimization has been the subject of studies done within a number of research and development projects. One of them was CoolEmAll [274, 299] which involved several authors of the current paper and investigated how cooling, heat transfer, IT infrastructure, and applications influence the overall energy and cooling efficiency of data centers. CoolEmAll provided advanced planning and optimization tools - SVD Toolkit (Simulation, Visualization and Decision support toolkit) enabling analysis and optimization of data center infrastructure in a holistic manner. The toolkit consists of several modules supporting IT infrastructure designers, decision makers and administrators in the process of planning new data centers or improving existing ones. In particular, the tools allow for detailed modeling of IT infrastructures, evaluating how they react to various load and power conditions and simulate the corresponding cooling facilities behavior.

The core of the SVD Toolkit is Data Center Workload and Resource Management Simulator (DCworms) [300] and the CFD simulation module. The former tool combines scheduling and management policies with power profiles and cooling models of data center infrastructure [275], while the latter one provides heat flow model supporting hot spots detection. In this way, SVD Toolkit allows evaluation of various arrangements and configuration of racks and cooling components until an optimal airflow through the server room is achieved. Models available within DCworms allow for estimating thermal load at different hardware levels, (ranging from single chassis, through the whole rack up to the data center) and analysis of various cooling approaches based on fans, chillers, computer room airside handlers and dry coolers (free cooling).

DCworms is still developed by Poznan Supercomputing and Networking Center and used for evaluation of energy efficiency and different cooling strategies within other, ongoing, EU funded projects and is available for use under open source-based license.

5.3.3.2 Thermofluid Simulations

One of the earliest uses of supercomputers was for thermofluid coupled simulations. There are a large number of software packages that can do thermofluid simulation . However, detailed simulations are computationally costly, therefore simpler simula-

tion models are often used. A modular approach is often used, with low level details from one simulation being fed into a larger less well resolved simulation.

For an air cooled system in a hot aisle cold aisle data center, one can start at the level of a simple rack in an air cooled cabinet. If the rack is operating at maximum capacity, then one can calculate the required air flow rate to keep the system temperature at a sustainable level. In such an application, the air flow speed will be much less than the speed of sound, and so the low speed compressible Navier Stokes equations or the incompressible Navier Stokes equations can be used. The dominant heat transport mechanism will be convection, so once the heat has been transferred to the fluid, it can be considered as a turbulently mixed passive scalar. The room can be initially modeled as a source of cold air and a sink for hot air. Thus a single simulation can be used to model all the racks in the data center. With this information, the required airflow in the room can be calculated using the low speed compressible Navier Stokes equations or the incompressible Navier Stokes equations with the Boussinessq approximation to allow for both free and forced convection and conduction.

Simulation of liquid cooled systems is more complicated because the details of heat transfer between different components is challenging. In cases where phase changes occur, accurate thermofluid simulation is extremely challenging and it may be best to use tables and correlation functions to estimate heat transfer rates. For liquid cooled systems with immersion, low speed compressible Navier Stokes equations or the incompressible Navier Stokes equations with the Boussinessq approximation can again be used. Accurate meshing of all the geometry is complicated, thus a simpler representative model of the heat generating units would likely be sufficient for design purposes. The heat exchanger also need not be fully simulated, provided it is adequately characterized, in particular by having some model for heat absorbed from the primary cooling fluid as a function of temperature and circulation velocity and heat transmitted to the secondary cooling fluid. Engineering tables can be used to aid building design and/or integration with the building heating/cooling system to ensure for adequate operating conditions.

Example thermofluid simulation packages include:

- OpenFOAM [301]
- OpenLB [302]
- Palabos [303]
- ANSYS Fluent [304]
- COOLSim [305]
- COMSOL Multiphysics [306]
- STAR-CCM+ [307]
- 6SigmaRoom [308]
- FloTHERM [309]
- TileFlow [310]

Specific data center simulation tools such as COOLSim, 6SigmaRoom, FloTHERM and TileFlow can make it easy to design a data center. Data center simulations are also good educational computational modeling projects. Typically simulations will

not include all rack details, but will instead consider a hierarchy of models. Thus experimental confirmation will be required to validate the calculations before using them for design.

The simulation tools can be used to ensure the operational environment will be suitable before deployment or upgrade of a data center. They can also be used in cost optimized design of a data center. Due to the long data enter building lifetimes, careful design choices are required since retrofitting costs for data center computer hardware upgrades should be minimized. Design optimization is an area that still needs to be fully applied to data centers, though it seems that some existing tools such as Dakota [311] for design optimization, can be coupled to the data center simulations, for example one can couple Dakota to OpenFOAM to optimize cabinet placement for air cooled data centers, or component placement in an immersed forced convectively cooled cabinet.

Modular pod data centers that are entirely self-contained are attractive for cloud computing applications [291]. Due to their limited size, airflow in these can be carefully optimized and validated to ensure sufficient cooling is obtained at minimal cost, provided the heat can be exhausted efficiently.

5.3.4 Future Challenges

In this chapter we have given an overview of liquid cooling strategies for data centers. One of the challenges that remains is to determine which of these strategies data centers follow and how effective they are. There are a number of data collection and dissemination activities related to high performance computing data centers, in particular VI4IO [312] which has as an additional component the comprehensive data center list and the Energy Efficiency working group [313], which is aiming to standardize liquid cooling deployments. It will be challenging to have plumbers and computer scientists work well together, but this will be required for effective data center design. A long lived repository with computer center cooling strategies is also of interest. Such developments would allow for the development of a shared body of knowledge on data center design and operation.

Of particular interest in a survey are:

- Data center location
- Data center layout
- Number of server rooms
- Characteristics of the server rooms
  - surface area
  - number of hosted racks/cabinets
  - CPU used and maximum operating watts
  - power distribution (number of PDU feeds etc.)
  - rack capacity (kW), type, vendor and usage (HPC, HPC hybrid (incl. accelerators), storage, interconnect...)
  - servers in a rack with particular cooling technologies
  - Computer aided design diagrams of the motherboard
    - airflow – also specify the type (hot/cold aisle...)

    - Indirect Liquid cooling (such as airflow with water-cooled doors)
    - Direct Liquid Cooling (DLC) – also specify the type (immersive, heat loop, phase change...)
  - expected/measured PUE (if measured, explain how):
  - inlet water temperature
  - ambient room temperature
  - outlet air/water temperature
  - Type of Heat recycling
    - building warming
    - electricity generation
    - none
    - other
- Heat exchanger (type(s) and capacity):
- Seasonal temperature variation accommodation:
- Fire Protection technology (argonite...)
- Idle power consumption
- Maximum power consumption
- Was simulation used to help design your data center and/or integrated cooling solution?

Our initial attempt at data collection for a small subset of data centers indicates that such a survey will be difficult to fully populate and establish reliable entries for, though would be useful for the environment and for the high performance, big data and cloud computing industries. The PUE is a simple metric to collect, but because simple well resolved component wise power measurement techniques are not yet part of many data center deployments PUE reporting is not always easy to obtain. Awards do seem to encourage and highlight innovation [263], but care is required to ensure they lead to long term best practices.

5.3.5 Conclusion

Some options and considerations for data center cooling have been presented. Data centers are expected to use upto 3% of all electricity generated worldwide. Depending on the data center, cooling can be upto 30% of the electricity costs for the data center. Thus a good cooling solution which has a high initial setup cost can pay for itself quite quickly. While data center design specifications have not fully standardized, open source hardware will force some standardization. Measurements of PUE, though not a good indicator on its own, has raised awareness of inefficient data center cooling methodologies that lead to high operating costs. A better measure of data center cooling effectiveness is needed, but may take time to develop. To do so, it is helpful to collect and curate information on cooling strategies used in different data centers.

Acknowledgments: We thank Natalie Bates, Jean-Jacques Chanot, Davide Marini, Alexander Moskovsky, Dale Sartor, Gert Svenson, Phil Tuma, Siddarth Venugopal, Jean-Marie Verdun, Andrew Winfer, Wilbert Yuque, Jason Zeiler, members of the Energy Efficiency Working Group https://eehpcwg.llnl.gov/ and members of the OpenCompute project http://www.opencompute.org/ for helpful discussions.

5.4 A Full-Cost Model for Estimating the Energy Consumption of Computing Infrastructures

Since its advent in the middle of the 2000's, the *CC* paradigm is increasingly advertised as a price-effective solution to many IT problems. This seems reasonable if we exclude the pure performance point of view as many studies highlight a non-negligible overhead induced by the virtualization layer at the heart of every Cloud middleware when subjected to an HPC workload. When this is the case, traditional HPC and Ultrascale computing systems are required, and then comes the question of the real cost-effectiveness, especially when comparing to instances offered by the Cloud providers.

Many public or private organizations have departments and workgroups that could benefit from HPC resources to analyze, model, and visualize the growing volumes of data they need to conduct business. From a general perspective, HPC is essential for increased competitiveness and stronger innovation. There are two realistic scenarios today to access HPC capacities beyond what is available from the desktop systems. One option is to acquire and operate an HPC system. However, for many companies and especially SMEs, this is seen as a non-viable solution since the Total Cost of Ownership (TCO) is perceived as too high, and additional skills and manpower are needed to operate and maintain such a system. With the rapidly growing enthusiasm around the *CC* paradigm, and more particularly of the *IaaS* model which is best suited for HPC workload, a second viable option is foreseen and attracts more and more attention due to the massive advertisement toward the cost-effectiveness of this approach. In this model, users rent to providers a *resource* that may be computing, storage and network or higher level services. At present, many large actors of the Cloud Computing market propose an IaaS service to their users. The cost model of IaaS is based on the actual resource usage of the user, who is thus billed depending on his activity. The computing resources are operated upon virtual machines and ran on a multi-tenant mode on the physical hardware. Characterized by their scalability and high-availability, IaaS platforms tend to be more and more efficient and are now sliding towards the territory of the traditional HPC facilities. Because of these interesting characteristics, many IaaS implementations have been largely studied and benchmarked in the context of an HPC usage.

While it is now widely established that running an HPC workload on top of IaaS resources induces a non-negligible performance overhead due to the hypervisor at the heart of every Cloud middleware, many people assume that this performance impact is counter-balanced by the massive cost savings brought by the Cloud approach. *But is it true?* Since 2007, the *UL* operates a medium size HPC facility [314] ($\simeq$ 10100 computing cores, 346 CPU TFlops and a shared storage capacity of 8.8 PB over parallel and distributed File Systems as of May. 2018) addressing the needs of academic researchers coming from many different fields (physics, material sciences, economy, life-science etc.). While developing our own expertise in the management of such a platform, it also gave us the opportunity to scrutinize on a daily basis and over a significant period of time its associated operating costs.

In this context, [315] proposed a TCO analysis for this in-house HPC facility for the time period 2007-2014 which is reported in the next sections. Also, although the comparative performance of Cloud vs. HPC systems received a wide audience in the recent literature, the analogous analysis covering the costs remains at an early stage, with very few contribution from the academic community. This is also due to the constant and rapid evolution of the Cloud instances offered by the different providers and the frequent price changes, making it hard to establish a fair comparison of the Cloud usage price with regards the equivalent HPC infrastructure operational costs. Furthermore the direct comparison of a given Cloud instance price with an HPC operating cost is biased and should be taken with precautions as it omits the performance aspect where the Cloud instance performance does not match with the one of an HPC node. To feed this gap and allowing a fair cost analysis, the approach proposed in [315] was twofold. First a theoretical price - performance model was established based on the study of the actual Cloud instances proposed by one of the major Cloud IaaS actors (Amazon). Then, based on the owned facility TCO and taking into account all the *OPEX*, an hourly price comparison is proposed.

Electricity is becoming a major expense in current Cloud and HPC data centers, even chasing the IT hardware cost [316]. Three main approaches can help reducing this energy consumption: improving the infrastructure's energy efficiency, exploiting renewable sources, and increasing the utilization rates. While the two first solutions partly rely on investments, the last one only depends on the facility's management policy. Energy-aware policies could rely on users willingness to decrease their electricity usage. Yet, users are unaware of the energy consumption induced by their resource's utilization. It is thus crucial to provide the energy models for data centers. This was covered in two other studies [317, 318] summarized in this chapter.

In this section, and inspired by the work proposed in [315], we propose a TCO analysis of an in-house academic HPC facility of medium-size (in particular the one operated at the University of Luxembourg since 2007, or within the Grid'5000 project [319]), and compare it with the investment that would have been required to run the same platform (and the same workload) over a competitive Cloud IaaS offer.

This rest of this chapter is organized as follows: Subsection 5.4.1 reports the TCO analysis conducted in [315]. Then, the refinement of the model specialized for energy cost modelization [317] with recent cooling technologies is depicted in the subsection 5.4.2. Finally, subsection 5.4.3 concludes the paper and provides some future directions and perspectives opened by this study.

5.4.1 Total Cost of Ownership analysis for a medium-size academic HPC facility

The *UL* operates since 2007 an HPC facility [314] and the related storage by a relatively small team of system administrators. The aspect of bridging computing and storage is a requirement of UL service – the reasons are both legal (certain data may not move) and performance related. Nowadays, people from three faculties as many interdisciplinary centers within the UL, are users of this facility.

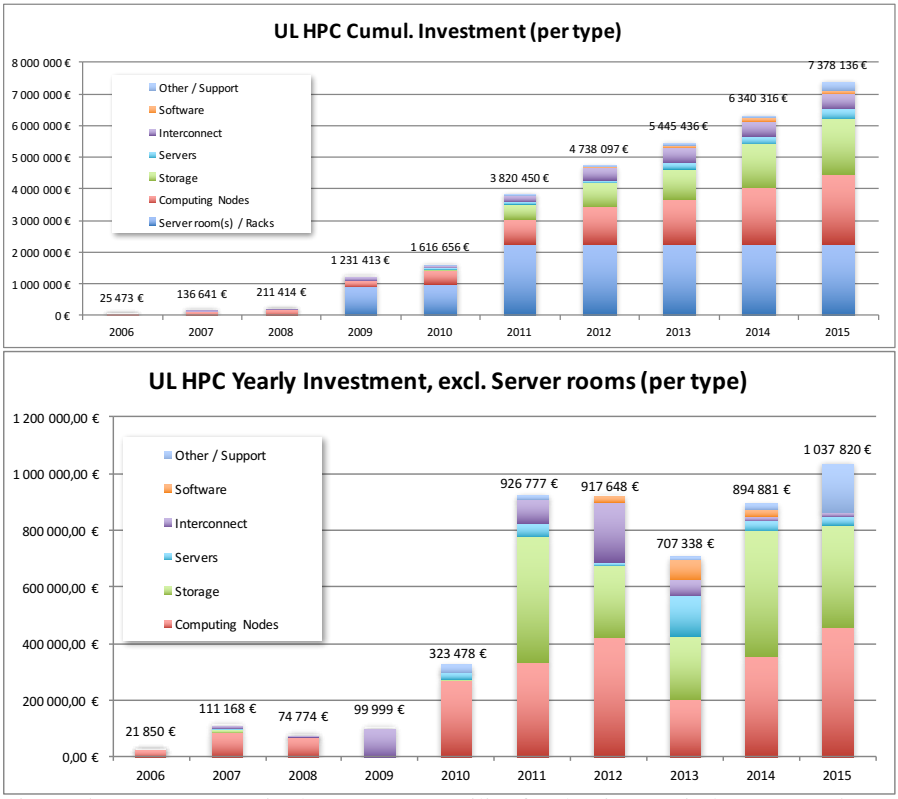


Figure 5.10: Investment in the UL HPC Facility for the time period 2007–2015. top: cumulative, incl. server rooms; bottom: yearly, excl. server rooms.

As of May 2018, it consists of 5 clusters spread across 2 geographic sites featuring a total of 662 computing nodes (for 10108 cores) and a cumulative shared storage capacity of 8.8 PB over parallel and distributed File Systems. In the reported study, the data and metrics collected are restricted to two clusters (namely *chaos*, *gaia*) that compose part of the UL HPC facility. Within these clusters, the internal interconnect is based on an Infiniband QDR (40Gb/s) network built on top of and adapted topology ("fat tree" and/or "star" depending on the situation) to ensure the best performances for the process communications. It is worth to note that such a technology out-performs by far all Ethernet-based network in use among major Cloud providers (including Amazon). A more recent cluster (*iris*) acquired on 2017 features up-to-date hardware (Infiniband EDR, broadwell and skylake processors) but is not part of reported study, which restrict to the time period 2007-2015. In all cases, the cumulative hardware investment has reached 6.34 M€ by the end of 2014 (7.38 M€ by 2015) and its evolution is depicted in the Figure 5.10.

The manpower is also a crucial piece of information often missing in many cost analysis. In the Table 5.7, these costs are highlighted in the case of our facility. They reflect a reality where despite all efforts, the local team managing the facility

Table 5.7 UL HPC Manpower.

Year	PM effort	Total Annual Cost
2006	1	11 553,63 €
2007	4	32 118,40 €
2008	19	64 200,24 €
2009	13	60 450,24 €
2010	9	57 864,00 €
2011	13	87 379,20 €
2012	25	136 947,68 €
2013	49	194 280,10 €
2014	40	173 026,90 €
2015	54	254 619,13 €

were outnumbered. While the TCO analysis proposed in this section relies on these numbers (more precisely, on the normalized prices for the year 2014), we estimate to 7.35 **FTE** (thus $\simeq$ 88 **PM**) the ideal manpower that would be required to maintain the platform at its sizing in 2015, in the repartition reported in the Table 5.8.

Another crucial question when performing a TCO analysis is the hypothesis linked to the usage of the platform. In our case, we have the chance to ensure a consistent and precise monitoring of the HPC resource usage (CPU, memory etc.) with regards the submitted jobs. For instance, a typical usage of the *gaia* cluster resources is illustrated in the Figure 5.11. It permits to derive an accurate ratio in terms of *used resources* with regards to the available computing components.

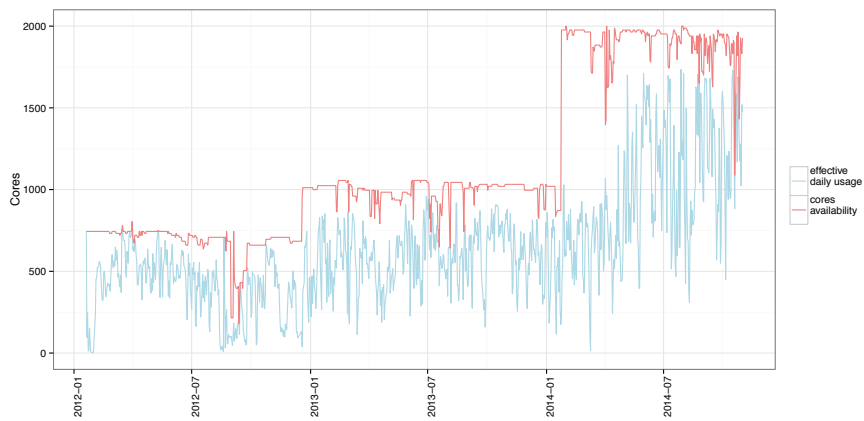


Figure 5.11: Example of used vs. available computing resources within the `gaia` cluster.

From this information, the TCO of the local HPC facility can be estimated, more precisely for the two clusters `chaos` and `gaia`. These clusters being heterogeneous and composed of different node classes as reported in Table 5.9, the hourly cost of a node belonging to each node class is computed accordingly and reported. In all cases

for the calculation of the TCO, we take into account the amortized node purchase price, network and server equipment price, room equipment, manpower and energy (power and cooling) costs. Computing nodes are considered amortized on a four year basis, network equipment on eight years and server equipment on three years. Server rooms on their side are supposed to be amortized in 15 years. Unfortunately the only thing that could not be integrated into this TCO analysis is the building cost itself as the HPC center is hosted within a university which, as a research institutions, benefit from governmental funding for its buildings. For obvious privacy reasons, the detailed buying price of the machines cannot be disclosed. Yet the final amortized node hourly costs are reported in the last column of the Table 5.9. To facilitate the comparison with Amazon EC2 instances prices, this TCO is reported in US Dollar rather than in Euro.

Amazon EC2 Instances for HPC and equivalent cost models

Since its launch in 2006 with the *m1.small* instance, Amazon has largely expanded the variety of instances and features it provides through its EC2 offer. They are now proposed in several families, each corresponding to a different computing need. Along the years and at new instance type releases, Amazon decreased the prices of the already existing ones.
To ensure backward compatibility, most older instances are still distributed, however they do not provide a good price/performance ratio anymore regarding the most recent ones.

Table 5.10 presents all the available instance types and their release dates as of early 2018 (except for the *cc1.4xlarge* which is not available anymore). An instance type e.g. *m1* or *c1* belongs to an instance family, e.g. *General Purpose* or *Compute Optimized*. For each instance type, there exist one or several models available that are not presented in this table. Instance models are described by an extension of the instance type (e.g. *m1.small*), possible values are: micro, small, medium, large, xlarge, 2xlarge, 4xlarge, 8xlarge. A given instance model corresponds to a particular performance of the instance in terms of vCPU, Memory, Storage, Network or such other performance characteristics.

To summarize, the cloud characteristics (which are also valid for EC2) that may impact HPC applications are the following:

1. Clouds are run in *virtualized environments*. Although the virtualization overhead is not so important for CPU bound applications, the virtualized networking is still an important drawback for applications that need communications (i.e a large part of HPC applications). Even though the networking uses virtualization improvements such as *SRIOV*, there exists a performance drop for inter-node communications.
2. There is generally no high performance network such as Infiniband yet available. This is still a problem for many applications whose performance is highly linked with the network performance.
3. Cloud by default uses multi-tenancy. This is a real problem as it does not ensure a reliable and stable behavior for applications. In particular, I/O operations in

multi-tenancy platforms can face high I/O jitter. However EC2 also provides a charged *Dedicated Instances* service to ensure the user will have the exclusive access to the nodes reserved.

4. Spatial and link proximity of the reserved VM is not guaranteed by default. Because of this intrinsic characteristic, inter-node bandwidth and latency can vary a lot. As for multi-tenancy, EC2 also provides for some instances the *placement group* feature, a logical grouping of instances within the same Availability Zone (isolated location within the same region) ensuring a low-latency, 10 Gb network between the nodes.

If we now aim at the execution of an HPC workload, one of the key constraint to address is the need of a reliable and efficient underlying network. Not all EC2 instances offer such a guarantee, however there exists a mechanism called *placement group* available for some of them and that allows the user to group a set of instances in the same cluster. This is an important requirement for HPC applications that use several nodes. Here we consider that most *classical* HPC applications fall into that category and thus need to be placed within the same cluster.

It is also important to say on an HPC point of view that the nodes that host the EC2 instances have HyperThreading activated, thus each vCPU on an instance is actually a HyperThread. In all cases, Amazon's EC2 provides several mechanisms to target High Performance on its cloud instances.

1. Enhanced Networking with *SRIOV*. This hardware feature, proposed on the most efficient instances, provides higher network performance for the VM: higher bandwidth and lower network latency and jitter. The instances were this feature is available are *c3*, *c4*, *r3*, *i2* and *d2*. Unfortunately GPU instances do not provide the Enhanced Networking feature but we have to consider them anyway for an HPC environment.
2. Placement Groups (or cluster networking). This feature provides high bandwidth performance with a full bisection Ethernet network.
3. Dedicated Instances. By default on EC2, cloud instances are multi-tenant VM hosted on the same physical node. By opting for the dedicated instance option (paying) at launch time, the instance will have exclusive access to the underlying physical machine.
4. EBS-Optimized. EBS is Amazon's persistent storage for EC2 instances (as opposed to instances local storage which is destroyed after the instance termination). It can be purchased with a provisioned IO option to increase and guarantee data transfer rates. EBS volumes have a maximum throughput of 128 MiB/s that can be attained only with instances proposing the EBS-optimized option.
5. One of the latest instance type released by Amazon at this time, the *c4* and *c5* instances, proposes EBS storage with no additional costs but does not provide local storage.

In practice, EC2 instance performances are provided in **ECU**. This metric represents the relative measure of the integer processing power of a given instance. Amazon does not give precisely how they measure ECU, the only information available is that one ECU is equivalent to one Intel Xeon core running at 1.0-1.2 GHz. It is also

clearly stated by Amazon that this metric might be revised in the future by adding or substituting measures that go into the definition of an ECU. According to Amazon, the goal of providing instance performance as ECU is to make it easier to compare CPU capacity between different instance types and thus to "*provide a consistent amount of CPU capacity no matter what the actual underlying hardware*". Although the claim of using several benchmarks and tests to manage the consistency and predictability of the performance from an EC2 Compute Unit, it seems that this information might be sometimes misleading. In [320], the authors showed that for several instance types the performance measured experimentally was unreliable, unpredictable and not reflecting the ECU provided by EC2. This was due to the fact that the instances that were tested in this study were running indifferently on several types of CPU and it seems that for instances whose CPU model is described as being a *Xeon Family*, there is no guarantee of the actual *CPU* model on which the instance will be ran.

In another study driven by Iosup et al. in [321], the authors experimentally compared the actual GFLOPS regarding the announced ECU for *m1* and *c1* instances and observed a real performance comprised between 39 to 44% of the advertised performance for *m1* instances and most of *c1* instances with an exception of 58.6% for the *c1.xlarge* instance. It follows that as ECU is a non standard way to measure a processor performance and it actually makes it difficult to fairly compare the computing performance of an EC2 instance with an HPC node. Moreover, as this metric is not provided with a strict definition and as it was observed performance mismatch between the announced ECU and actual performance we need another metric such as FLOPS which is the traditionally used theoretical processor computing performance metric. As according to its definition, an ECU is equal to one Xeon core at 1.0-1.2 GHz, the ECU should reflect the processor FLOPS value.

In [315], the authors offered an accurate linear model exhibiting the relationship between ECU and GFLOPS. Based on multiple linear regression, an automated step-wise selection was first performed. Then, from the meaningful parameters detected, a manual assessment of the ones that are the most representative in the model via R^2 shrinkage is executed. Among many parameters evaluated, we established that the significant ones are: Processor speed (GFLOPS), Memory size (GB), Disk Size (GB) and number of GPU cores. It follows that the new price model for EC2 instances prices can be described through the following equation:

$$Instance\_Price = \begin{cases} \alpha * GFLOPS \\ +\beta * Memory\_GB \\ +\gamma * Disk\_GB \\ +\delta * Nb\_GPUs \end{cases}$$

It appears that instances that were released in the same time period could be grouped in the same price model without changing significantly the fitting – this assessment would probably need to be checked against recent instances (*i.e.* after 2016). Table 5.11 provides the parameters for the refined On Demand pricing models obtained for five successive *generations* (*i.e.* release period) of Amazon instances.

These five models have been evaluated individually, leading to very low error rates, or even perfect linear fitting [315].

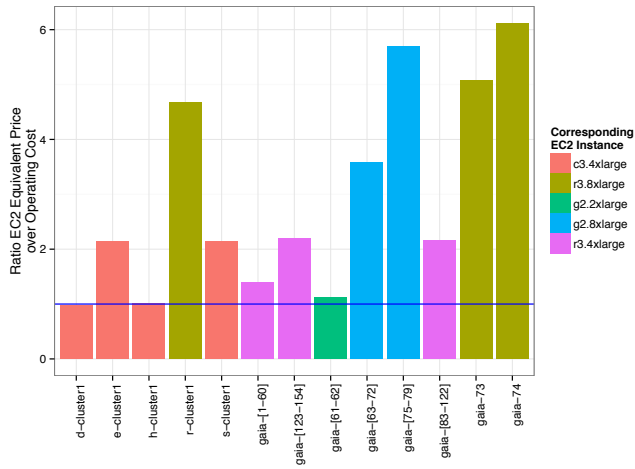


Figure 5.12: Ratio between the EC2 Equivalent Price of the UL in-house HPC Resources with regards their respective TCO estimation.

From this price model, it is possible to provide accurate EC2 equivalent prices for a given class of node within a local HPC facility. This means that even though not all the cluster computing nodes have a perfect cloud instance match, we are still able to determine what would be its equivalent price on EC2 if that matching instance was available. This information is later used to assess the interest of operating a given node class regarding renting an On Demand instance with the same performance on the cloud. Such an estimation is proposed in Figure 5.12 which presents the ratio, for each node class in the cluster, of the *EC2 Equivalent Price* based on the closest possible existing EC2 instance, versus the operating cost of the considered node as evaluated previously (see Table 5.9). It can be observed that for a few node classes, the ratio is close to 1, meaning that operating in-house these HPC nodes costs about the same price as renting the corresponding instance on Amazon. Thus it may be interesting to rent these kind of resources instead of operating them locally. However for *all the other ones*, renting cloud instances is simply too costly. Let's also remind that the operating cost provided in Table 5.9 represents the *maximum* cost scenario where the HPC infrastructure is used at 100%. For a lower utilization of the cluster, this operating cost will be decreased due to lower energy usage (see also the section 5.4.2).

Finally, the pricing model can be extended as the above comparison is based on "On Demand" instance prices. In this purchase model, the customer pays a fixed rate by the hour with no commitment. In this scenario, the acquisition of an in-house HPC facility is probably a more cost-effective solution. However, cloud provider such as Amazon proposes cheaper options for the instance prices, in particular through *Reserved* instances which offer capacity reservation and a significant discount on the hourly charge of a given instance. There are three payment options in this case: (1) *No Upfront*: this option provides access to a Reserved Instance without

requiring an upfront payment, (2) *Partial Upfront*: this option requires a part of the Reserved Instance to be paid upfront and the remaining hours in the term are billed at a discounted hourly rate, regardless of usage. (3) *All Upfront*: a full payment is made at the start of the term, with no other costs incurred for the remainder of the term regardless of the number of hours used.

It permits to investigate how these instance prices impact the running costs on a yearly basis. For that purpose, all job requests scheduled on the UL HPC platform (as extracted from the logs of the batch scheduler) have been analyzed on a reference time period (the year 2014 in this case). The results are presented in the Figure 5.13. It may seem surprising at first that the cost of operating the cluster locally is 40% cheaper than renting on EC2 with the all upfront reserved mode and 2.5 times less expensive for on demand pricing. Actually, a large part of the workload from *chaos* and *gaia* use nodes with a low local operating cost compared to their *EC2 relative price*. This result particularly shows that the migration of HPC workloads to the cloud is not only a problem of adaptability of the cloud performance to HPC needs but also a problem of correctly determining which types of job are good candidates to be run on the cloud in order to prevent such cost overheads.

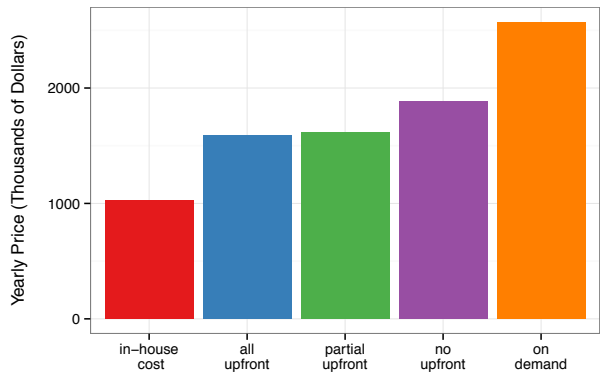


Figure 5.13: Annual cost for year 2014 of UL workload if operated locally or rented on a cloud provider (Amazon EC2).

5.4.2 Energy cost models for Clouds

Electricity lies among the costs incurred by a data center and this cost increases with the size of the data center. While it could be optimized with the help of users, they are often unaware of the actual energy consumption of their virtual resources. Indeed, the cost of renting a VM on a Cloud like EC2 (as presented on Table 5.10) depends mainly on the VM size and the resources' usage within the VM does not influence its price. Energy cost models are needed to increase users' energy-awareness and to provide an incentive towards a greener behavior.

The power consumption of a given device (computing, networking or storage) is divided into two parts: a static part that corresponds to its energy consumption

while being powered on but idle, and a dynamic part that changes with the workload. Servers are not yet power proportional: their static power consumption is high and the dynamic part does not depends linearly on the workload [322].

Figure 5.14 shows the power consumption of a synthetic workload running on a server (Nova node, detailed later on Table 5.12). The idle power consumption of this server is around 70 Watts.

Figure 5.14: Power consumption of a server running a synthetic workload

Furthermore, devices are heterogeneous within the same data center as detailed for servers in Table 5.9. An energy cost model for VMs needs to consider both static and dynamic parts in order to discourage resource overprovisioning (costly due to the non-proportionality) and to promote dynamic energy-efficient techniques (such as Dynamic Voltage Frequency Scaling) respectively.

In addition to the electricity consumed by ICT resources, data centers contain other devices consuming electricity for a proper functioning of the infrastructure: cooling systems, power delivery components and other miscellaneous elements such as the lighting. The deployment of VMs indirectly causes this additional consumption. Consequently, it should be included in the VM energy cost model.

Finally, when requesting VMs, users receive access to virtualized resources. Concretely, it implies that users may share physical resources and as previously explained, this sharing, combined with the virtualization layer may induce performance degradation and an increased energy consumption. A fair accounting is needed to attribute each Watt consumed by the data center to a user, including the Watts that cannot be measured by a software wattmeter that would monitor the VM (i.e. the indirect costs).

In order to be fair among users, an energy cost model for VMs needs to consider:

- the size of the VM to account for the booking of hardware resources
- the heterogeneity of servers, but the cost for a given VM should be independent from its physical allocation on a server (as users do not decide this allocation)
- the network and storage devices
- the non-IT equipment of the data center that consumes electricity (including cooling devices)

An energy-propotional model is proposed in [317]. This model is named EPAVE: Energy-Proportional Accounting in VM-based Environments. It consists in a power-aware attribution model for VMs taking into account the overall consumption of the data center and considering the heterogeneity of servers and VMs. To account for non-IT equipment, it uses the Power Usage Effectiveness (PUE), a widespread metric for measuring the infrastructure's energy efficiency for data centers. The PUE is given by the ratio between the overall data center power consumption over the power consumption of IT equipment (i.e. network devices, compute and nodes). The closer it is to 1, the more efficient is the data center infrastructure. It highlights the consumption impact of cooling system, Power Distribution Units and other non-IT systems.

EPAVE comprises two parts: a static and a dynamic part. The static part deals with the idle consumption of servers hosting VMs, the power consumption of other IT equipment (including storage nodes and network devices) and the PUE.

We consider the Lyon site of Grid'5000 [319] as a use case to compute the energy cost given by EPAVE. Table 5.12 details its servers' characteristics: it consists of three clusters including 23 Nova nodes, 16 Taurus nodes and 79 Sagittaire nodes. Based on their idle power consumption and their number of cores, we can compute the static part of EPAVE, which consists in the average static power consumption per core in the data center. The dynamic part is computed by using the maximal power consumption of each server.

Figure 5.15 presents the energy cost provided by EPAVE only for the server part depending on the size of the considered VM (from 1 to 8 virtual CPUs) using the server configurations presented in Table 5.12. It also displays the actual power consumption of each type of servers. We can observe that the server heterogeneity, and especially Sagittaire servers that are numerous, old and inefficient, causes a consequent increase in the static energy cost.

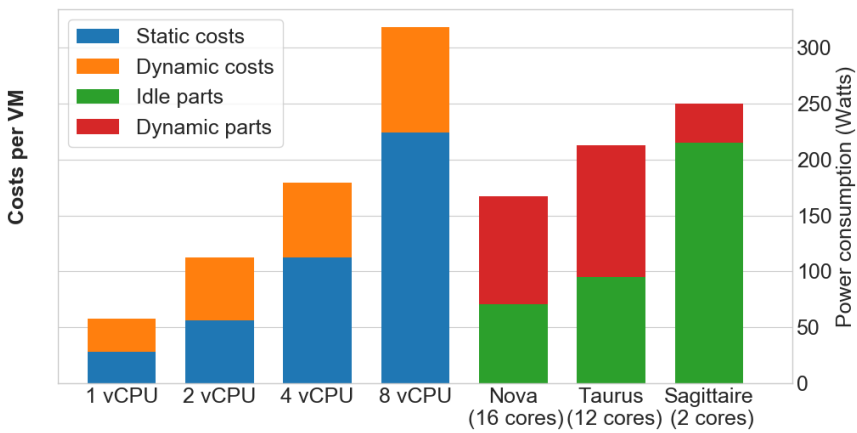


Figure 5.15: EPAVE cost model for servers only based on the Grid'5000 Lyon site

We consider that the electricity cost is around $0.2 per kWh in France from [323] and we use a PUE of 1.53 for the data center (PUE measured on Rennes site of

Grid'5000 [324]). The full static costs (with PUE, network and storage devices) are fairly distributed among the cores of the data center. It means that the energy cost determined by EPAVE for a VM with 1 vCPU hosted in this infrastructure is $0.02 per hour, so around $175 annualy. This amount is similar to what can be found in [316].

If we consider the first node listed in Table 5.9, it is estimated to have an evaluated hourly TCO of $0.386 for 12 cores. In terms of performances, this node is really close to the Taurus nodes considered in our platform (Table 5.12). It means that if we consider this node to belong to our data center, its induced electricity cost (according to EPAVE) would represent around 60% of its overall TCO cost.

While this model can be an incentive for users to consume less energy, it does not take into account the electricity source. Most renewable energy sources (e.g. sun and wind) are intermittent and variable by nature. For favoring renewable energy sources, instead of computing a financial cost, we propose a CO2-related cost [318].

This CO2 emissions accounting framework gives flexibility to the Cloud providers, predictability to the users and allocates all the carbon costs to the users [318]. Similarly to EPAVE, it considers the indirect costs induced by the userâĂŹs utilization, like for instance, the consumption of the data center air conditioning system. These indirect costs are fairly split among users over long time periods in order to ensure their predictability: they are provided to the users upon submission of their VMs requests. Finally, the electricity consumption is translated into CO2 emissions based on the provided electricity mix.

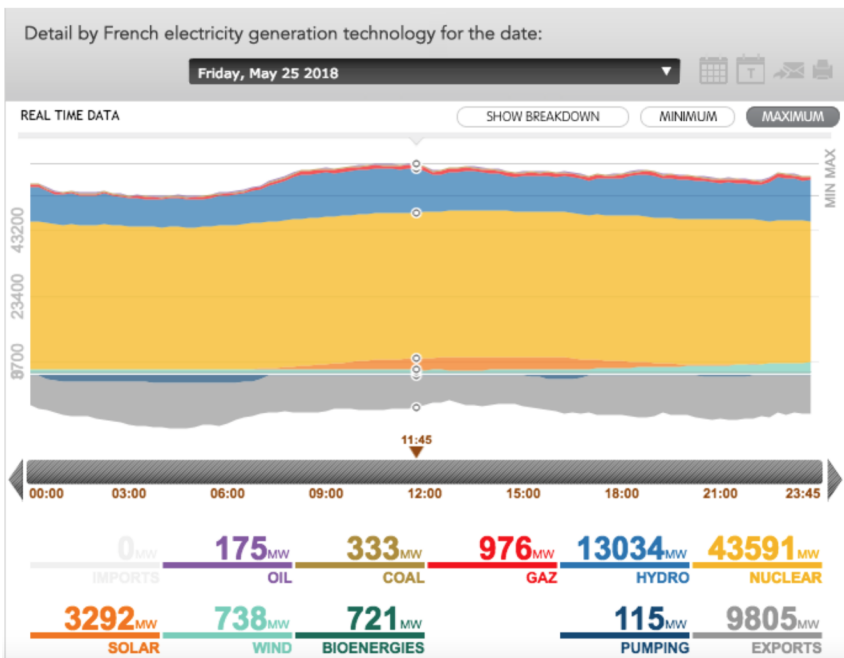


Figure 5.16: Example over a day of the energy mix in France provided by RTE [4]

For the CO2 computation, we rely on data provided by RTE, the French transmission system operator about the energy mix for France [4]. An example is provided on Figure 5.16 for one day (Friday 25th of May 2018). Current production status is updated every 15 minutes. RTE also provides the CO2 emissions per kWh of electricity generated in France per day. On this day, it represents on average 26 g/kWh. This figure is low due to high dependence of the French electricity mix on nuclear power.

It means that for our 1 vCPU VM, it represents approximately 2.5 g of CO2 generated per hour only for the utilization of this VM. This cost does not include the CO2 emissions due to the manufacturing of data center devices.

In this section, we have pictured an overview of two energy costs models for VMs: EPAVE, which provides an energy-proportional accounting in data centers [317], and a CO2 emissions accounting framework [318]. Both aim at fairly distributing all energy costs to the users, including indirect costs like the electricity consumption of cooling systems. They allow transparent, reproducible and predictive cost calculation for users and providers. The two models indicates that electricity costs are of uttermost importance when considering TCO of a data center.

5.4.3 Conclusion

There are two realistic scenarios today to access HPC capacities beyond what is available from the desktop systems. One option is to acquire and operate an HPC system, the other option relies on the Cloud Computing paradigm, and more particularly of the IaaS model which is best suited for HPC workload. In this context, we have proposed a Total Cost of Ownership analysis for a medium-size academic HPC facility and we compare it to the Amazon EC2 cost model.

The costs of current data centers are mostly driven by their energy consumption specifically by the air conditioning, computing and networking infrastructure. Yet, current cost models are usually static and rarely consider the facilities'ĂŹ energy consumption per user. Current pay-as-you-go models of Cloud providers allow users to easily know how much their computing will cost. However, this model is not fully transparent as to where the costs come from (e.g., energy). In this section, we have explored two full-cost models for estimating the energy consumption and the CO2 emissions of virtual machines hosted in data centers.

5.5 Heterogeneous computation of matrix products

Matrix multiplication is one of the most essential computational kernels used in the core of scientific applications. An application is, for instance, the calculus of matrix polynomials. Matrix polynomials are used, e.g. for the computation of functions of matrices such as the exponential of a matrix by the Taylor method [325]. Matrix functions appear in the solution of many engineering and physics phenomena, which are governed by systems of linear first-order ordinary differential equations with constant coefficients [326], control theory [327], or theory of multimode electric power lines [328]. Some other engineering processes are described by second order

differential equations, whose exact solution is given in terms of the trigonometric matrix function sine and cosine [329, 330], for which efficient solutions have been proposed in [331] and [332].

Matrix multiplications has been highly studied in the past in order to improve the efficiency of its computation in both sequential and parallel computer architectures. This operation has also received full attention in parallel heterogeneous environments. Some proposals use one of more GPUs in a node to improve the execution of matrix multiplications in order to accelerate applications like, for instance, the computation of the cosine of a matrix [333, 334]. Many contributions that try to use all the devices of a heterogeneous environment basically propose irregular partitions of the factor matrices that can efficiently be mapped on the computing resources; see for instance [335, 20, 336]. However, it is difficult to find actual implementations of the matrix multiplication on heterogeneous nodes that feature very different devices. The MAGMA project , for instance, aims to develop a dense linear algebra library similar to LAPACK but for heterogeneous/hybrid architectures; it is one of the most active projects that implement BLAS routines for nodes featuring accelerators [337]. Currently, MAGMA implements a version for NVIDIA GPUs in which the matrix multiplication is carried out only by the GPUs, i.e. the CPU does not intervene. The MAGMA project also provides with a version, MAGMA MIC, which provides hybrid algorithms that involve the host CPU and one or more Intel Xeon Phi processors. However, this project does not use both NVIDIA GPUs and MICs processor all together in the same host. Authors of [338] propose a programming model for heterogeneous computers featuring CPU, a GPU and a Xeon Phi with the aim to incorporate it to MAGMA library. However, they have not shown its proposal with matrix multiplication.

The work presented here is an experimental study of performance in execution time and energy consumption of matrix multiplications on a heterogeneous server. The server features three different devices: a multicore CPU, an NVIDIA Tesla GPU, and an Intel Xeon Phi coprocessor. Our implementation is based on a simple implementation using OpenMP sections. We have also devised a functional model for the execution time and energy consumption that allows to predict the best percentage of workload to be mapped on each device.

The rest of the section is structured as follows. Section 5.5.1 shows the application implemented to carry out a square matrix multiplication on these three different devices. The following section shows experimental results both in time and energy consumption of our application. The section finishes with some conclusions.

5.5.1 A Hybrid Matrix Multiplication Application

In order to solve efficiently problems that intensively use matrix multiplications on a heterogeneous server, we propose an implementation for a matrix multiplication application and present an experimental study of its performance in both execution time and energy consumption.

5.5.1.1 The Hardware and Software Used

The heterogeneous server we have been working with features the following devices:

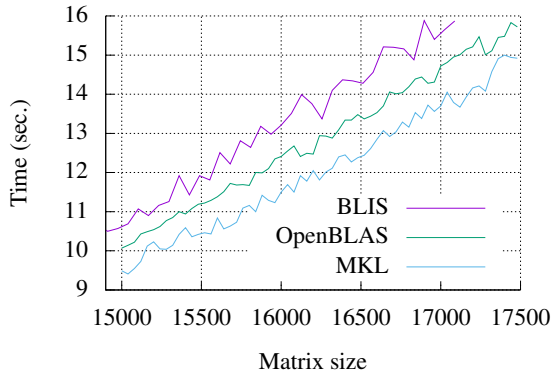


Figure 5.17: Execution time for a square matrix multiplication in one core using different libraries.

- **CPU**: Two sockets with an Intel Xeon CPU E5-2670 v3 at 2.30 GHz each. This processor has 12 cores so the server contains a total of 24 threads. The main memory of the host is 64 GB.
- **GPU**: NVIDIA Tesla K40c with 2880 cores and 12 GB of device memory.
- **PHI**: An Intel Xeon Phi 3120A coprocessor with 57 processors (228 cores) and 6 GB of device memory.

Although the term "device" is usually assigned to accelerators only, i.e. GPU and PHI, in the next and for the sake of simplicity, we will use it to denote the three of them.

On the software side, we have within reach different implementations of BLAS [339] to perform the matrix multiplication:

MKL: Intel Math Kernel Library is an optimized implementation of linear algebra routines contained in BLAS and LAPACK, and other mathematical functions like the FFT. This library is available for multicore x86 processors, and also for the Intel Xeon Phi coprocessor [340]. There exist "threaded" routines, e.g. the matrix multiplication routine `GEMM`, for both devices.

OpenBLAS: OpenBLAS is an optimized BLAS library based on GotoBLAS2 1.13 BSD version [341]. Used in the CPU.

BLIS: This library is self-described as "a portable software framework for instantiating high-performance BLAS-like dense linear algebra libraries". In addition the "framework was designed to isolate essential kernels of computation that, when optimized, immediately enable optimized implementations of most of its commonly used and computationally intensive operations" [342]. It has been used in the CPU.

CUBLAS: BLAS implementation for NVIDIA GPUs [343].

We performed a simple experimental analysis of the speed of the matrix multiplication (`GEMM`) in the CPU (Figure 5.17). For this test we used the maximum

available CPU cores, i.e. 24. (We ignored the fact that Hyper-Threading (HT) can be enabled to give a total of 48 logical processors. We observed that using just one thread per core is enough to fully exploit the execution resources of the core and not increase in performance can be achieve by activating HT.) It must be said that the performance of BLIS could be probably better by selecting the best parallel configuration. Contrary to the other two packages, BLIS is tuned by setting the value of up to four environment variables. That value corresponds to the number of threads that will be used to parallelize a given loop among the five nested loops in which the matrix multiplication is implemented in order to exploit the hierarchical set of intermediate memories of the most current architectures. In this test, only the outer loop was parallelized. A more suitable combination of values are likely to produce a better performance of BLIS, however, we decided not to test the large set of different combinations with the idea that barely the performance would outperform MKL in this machine. Consequently, we consider the performance of Intel MKL to be the best and, therefore, it is the only library used on the CPU side.

5.5.1.2 Implementation option

To proceed towards a heterogeneous matrix product, we started by implementing an application that partitions the problem into three concurrent pieces so that the three devices can cooperate in the solution. There exist different options to implement such an application. However, all the options can be gathered into two main classes standing for the use of light processes (threads), or heavy processes. The last option can be implemented e.g. by using MPI [344]. Here, we decided to use a simple approach based on threads, which are spawned by means of OpenMP subsections.

The application has been implemented with OpenMP subsections, so that each device code is included in a given subsection (Listing 5.1). The code for the Intel Xeon Phi, in lines 31–32, is implemented in a different source file (Listing 5.2) and compiled separately. This is because it is necessary to compile this code with the Intel C compiler (`icc`). For the compilation of the rest of the C code of the application we used the GNU compiler (`gcc`) since there exists incompatibility between the available versions for the NVIDIA compiler (`nvcc`, version 7.5) and for the Intel compiler (`icc`, version 16.0).

The basics of the heterogeneous multiplication are easy. To perform the multiplication $C = AB$, matrix A is completely broadcast to the two accelerators from the Host computer. Matrix B, however, is partitioned into three blocks of consecutive columns. The second block is uploaded to the GPU, the third one is uploaded to the PHI, and the first one remains into the host memory. The amount of columns of each block is denoted in Listing 5.1 by the values of variables `gpu_n`, `phi_n`, and `cpu_n` for the GPU, the PHI, and the CPU, respectively. Currently, the application receives these values as arguments by command line, in particular, the user sets the percentages for the GPU and for the PHI in the range $[0,1]$, the rest is computed by the CPU. Upon termination of the execution, the resulting matrix C appears partitioned and distributed among the three devices. We include in the application, and in the time measurement, the operation of gathering the result in the memory location allocated into the host to store the resulting matrix.

Listing 5.1: Code for the heterogeneous matrix multiplication.

```
int gpu_n  = (int) (gpu_weight * n);
int phi_n  = (int) (phi_weight * n);
int cpu_n  = n-gpu_n-phi_n;
#pragma omp parallel sections num_threads(3)
{
#pragma omp section
{ // GPU
  if( gpu_n ) {
  cublasHandle_t handle;
  CUBLAS_SAFE_CALL( cublasCreate(&handle) );
  double *gpu_A, *gpu_B, *gpu_C;
  CUDA_SAFE_CALL( cudaMalloc((void **) &gpu_A, n*n*sizeof(double) ) );
  CUDA_SAFE_CALL( cudaMalloc((void **) &gpu_B, n*gpu_n*sizeof(double) ) );
  CUDA_SAFE_CALL( cudaMalloc((void **) &gpu_C, n*gpu_n*sizeof(double) ) );
  CUBLAS_SAFE_CALL(cublasSetMatrix(n, n, sizeof(double), A, n, gpu_A, n ));
  CUBLAS_SAFE_CALL( cublasSetMatrix( n, gpu_n, sizeof(double),
                                     &B[n*cpu_n], n, gpu_B, n ) );
  CUBLAS_SAFE_CALL( cublasDgemm(handle, CUBLAS_OP_N, CUBLAS_OP_N, n, gpu_n,
                       n, &alpha, gpu_A, n, gpu_B, n, &beta, gpu_C, n ) );
  CUBLAS_SAFE_CALL( cublasGetMatrix( n, gpu_n, sizeof(double), gpu_C, n,
                                     &C[n*cpu_n], n ) );
  CUDA_SAFE_CALL( cudaFree(gpu_A) );
  CUDA_SAFE_CALL( cudaFree(gpu_B) );
  CUDA_SAFE_CALL( cudaFree(gpu_C) );
  CUBLAS_SAFE_CALL( cublasDestroy(handle) );
  }
}
#pragma omp section
{ // PHI
  if( phi_n ) {
  gemmPHI( n, phi_n, n, alpha, A, n, beta, &B[n*(cpu_n+gpu_n)], n,
           &C[n*(cpu_n+gpu_n)], n );
  }
}
#pragma omp section
{ // CPU
  if( cpu_n ) {
  dgemm( &transa, &transb, &n, &cpu_n, &n, &alpha, A, &n, B, &n,
                                                   &beta, C, &n );
  }
}
}
```

The code for the execution in the GPU is quite regular (Lines 7–27). It includes creation of the CUBLAS context, allocating memory for the three matrix factors, uploading matrices, executing the matrix product, downloading the result, and freeing the resources involved in the computation.

For the Xeon Phi, we used the "offload mode" of computation, that is, data is explicitly uploaded to the device and the operation is also explicitly executed there. Thus, the programmer have control of what exactly is executing the coprocessor. Arguments `in`, `out`, and `inout` specify clearly the direction of variables characterized

Listing 5.2: Code for the heterogeneous matrix multiplication in the Xeon Phi (offload mode).

```
void gemmPHI( int m, int n, int o, double alpha, double *A, int lda,
              double beta, double *B, int ldb, double *C, int ldc ) {
#pragma offload target(mic) in(m,n,o,alpha,beta,lda,ldb,ldc) \
                in(A:length(m*o)) in(B:length(o*n)) inout(C:length(m*n))
{
  cblas_dgemm(CblasColMajor, CblasNoTrans, CblasNoTrans, m, n, o,
                             alpha, A, lda, B, ldb, beta, C, ldc );
}
}
```

by those words. The operation is actually performed by calling to the BLAS matrix multiplication routine using the MKL version.

Finally, the code executed by the CPU only includes a call to the `gemm` routine (lines 38–39) for the matrix computation using MKL as well. We used the `fortran` interface instead of the C one used for the PHI for no specific reason but the application is oblivious of this.

Attention must be paid to the way in which the application is executed in our heterogeneous server. As it has been implemented, only three OpenMP threads are created so that each one will execute a different subsection. There will be, thus, one thread bound to each accelerator for data transference and control purposes. For the CPU case, however, the execution of the MKL routine will use only one thread. To use more threads (cores) collaborating in the matrix multiplication on the CPU side, the "nested parallelism" ability must be explicitly set. In addition, there are more environment variables that control the behavior of the application (Table 5.13).

This is an example of execution:

```
shell_$ MKL_NUM_THREADS=22 OMP_NESTED=TRUE MKL_DYNAMIC=FALSE
        MKL_MIC_ENABLE=0 MIC_ENV_PREFIX=MIC
        MIC_KMP_AFFINITY=balanced,granularity=fine
        MIC_OMP_NUM_THREADS=228 MIC_USE_2MB_BUFFERS=64K
        numactl --physcpubind=+0-11,12-23 program 10000 0.48 .15
```

The example executes the program `program` which generates two random matrices of order $n = 10000$. The GPU performs 48% of the computation, 15% is carried out by the PHI, and the rest, 37%, is computed by the CPU.

It should also be noted that the server has the hyperthreading enabled, but we decided not to use all the 48 threads and always use 24 as a maximum number of threads instead. For instance, when operating with the three devices, two threads are bound to one accelerator each, leaving the other 22 for the execution of the matrix multiplication in the CPU.

In addition, we have always used core affinity. This is to prevent threads from leaping amongst the cores at runtime, so as to reduce the variability of the execution times and also to improve the performance of all the devices attached to the host. Concretely speaking, we use the tool `numactl` to bind threads to cores.

The following is an example of the output of the application:

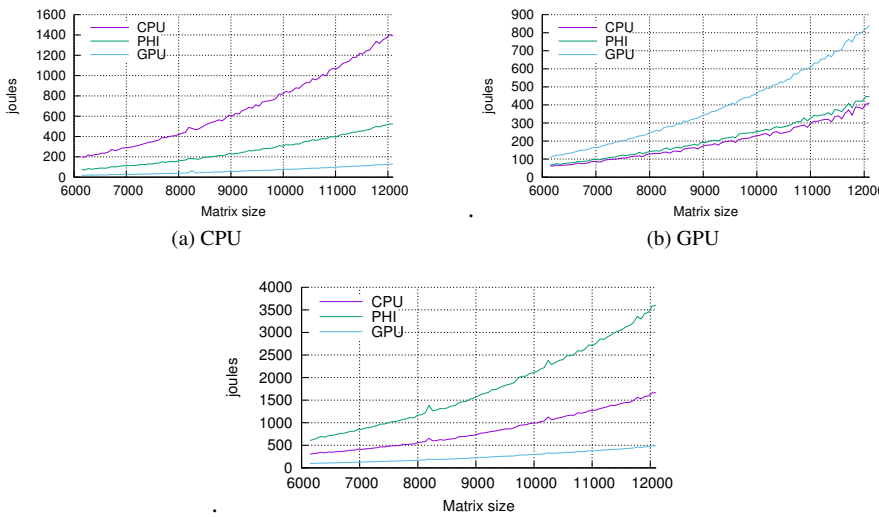
```
n = 10000 (CPU = 38.00
          (cpu = 1.17 sec. gpu = 1.14 sec. phi = 1.19 sec.)
          (1.30 sec. 1541.47 gflops)
```

for a random matrix of size $n = 10000$. The weight used for each device in this example results in a workload rather well balanced.

5.5.1.3 Energy consumption

We are also interested on evaluating the energy consumption of the devices participating in the matrix multiplication with the aim at, first, understanding the power trace of each device and, second, exploring a workload distribution which can result in energy savings.

For the energy measurement, we have used a tool called `powerrun` [345]. This tool is a wrapper to other tools for measuring the power draw of the CPU (uses PAPI and Intel PCM), of the GPU (uses NVML [346]), and of the PHI (uses Intel's MPSS [347]). The tool gathers the power samples of all the devices under operation and dumps a power trace to a file to compute the energy consumed during the execution time. This tool provides a library to instrument the code under test with simple calls that frame the part of the code to be measured.



(a) CPU (b) GPU

(c) PHI

Figure 5.18: Energy consumption when executing a matrix multiplication in the (a) CPU, (b) GPU, or (c) PHI while the other two devices remain idle.

We provide here three tests that show the energy consumption (in joules) of the three devices, respectively. In each test, the three devices are operating concurrently. Only one of them is working on a matrix multiplication while the other two remain idle. The test samples the energy of the three devices.

Figure 5.18 (a) shows the energy consumed by the system when only the CPU is "working" and is rather easy to interpret. The CPU is the most consuming device since it is the only one that performs useful work, while the other two consume the energy in idle state. It is also quite clear the difference in energy consumption when idle between the two accelerators, being very low in the case of the NVIDIA GPU compared with the Intel Xeon Phi.

Figure 5.18 (b) shows the energy consumption when the GPU is the only device operating on a matrix multiplication. Note that one of the cores of the CPU is also working since it is in charge of sending the two matrices to be multiplied and receiving the resulting one. The consumption of the Intel Xeon Phi is very large in idle state when compared with the CPU.

As expected, the consumption of the Intel Xeon Phi is quite large when executing the matrix multiplication (Figure 5.18 (c)). Also in this case one of the cores of the CPU is working to feed the coprocessor with the two factor matrices and to receive the solution matrix.

5.5.2 *Experimental Results of the Matrix Multiplication Application*

Figure 5.19 and Figure 5.20 show the execution time in seconds spent by the application to perform a matrix multiplication of two square matrices of sizes $n = 8000$ and $n = 14000$, respectively. The two graphics show times for different weight combinations. The percentage of computation carried out by the GPU is shown on the y-axis, while the work done by the PHI is shown on the x-axis. These two values are selected by the user. The rest of the computation is performed by the CPU. The figure shows less execution times (clearer cells) within the region between $\approx 25\%$ and $\approx 50\%$ for the GPU, and $\lessapprox 20\%$ for the PHI in the case of the problem sizes selected. There exists more opportunity for the PHI to participate as long as the problem size increases.

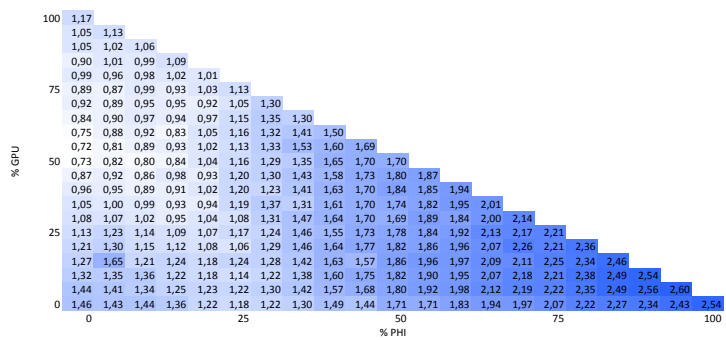


Figure 5.19: Execution time in seconds for a matrix product of size $n = 8000$ varying the weight of workload on each device.

Figure 5.21 shows the percentages of the minimum values obtained for the problem sizes $n = 8000, 10000, 12000, 14000$, which are 0.72 sec., 1.30 sec., 2.08 sec., and 3.20 sec., respectively. For large problems both the CPU and the GPU

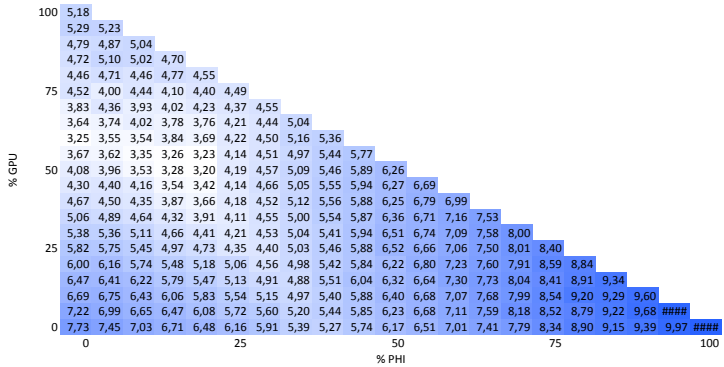


Figure 5.20: Execution time in seconds for a matrix product of size $n = 14000$ varying the weight of workload on each device.

reduce their weight to make room for the PHI, which does not contribute to the task with any size smaller than $n = 12000$. We can approximate the weight of each device, i.e. w_{cpu}, w_{gpu}, and w_{phi}[20], by the two linear functions shown in Table 5.14 for two intervals. By means of a larger experimental setup we could easily devise a functional model that allows to predict the best percentage of workload to be mapped on each device. However, we must take into account that there exist a problem size not very much smaller than $n = 2000$ for which it is not worthwhile to use the GPU. Also, for problem sizes $n > 14000$, the weight to be assigned to each device stabilizes around a fix value ($w_{\text{phi}} \approx 15\%$ and $w_{\text{ghi}} \approx 55\%$). However, as the problem size increases a little more, out-of-core algorithms are required and these functional models can significantly change.

Things are slightly different when we observe the total energy consumed by the matrix multiplication application. The minimum values of energy (in joules) are 379, 700, 1177, and 1783, for the problem sizes 8000, 10000, 12000, and 14000, respectively. Figure 5.22 and Figure 5.23 show, as an example, the energy consumption with problem sizes $n = 8000$ and $n = 14000$, respectively. The corresponding weights of w_{gpu} in which we can find these minimum values are 55%, 60%, 60%, and 60%, for each problem size, respectively, and 0% for w_{phi}. These numbers show that while Intel Xeon Phi can contribute a little to reduce the execution time, it can contribute nothing towards reducing the energy consumption in any case, as it was expected according to Figures 5.18 (a)–Figure 5.18 (c). The NVIDIA GPU is, currently, a more efficient device for HPC. In this particular server and for large problem sizes ($n > 10000$), the Intel Xeon Phi is used as a trade-off between execution time and total energy consumption.

We also figured out the dynamic energy of the application, i.e. the energy due to the execution of the application and that we obtain after taking away the energy consumed by each device in idle state. The results showed that we can find the minimum value for the dynamic energy for all problem sizes when none of the

[20]Note that the number of matrix columns assigned to a device d is $n_{\text{d}} = n \cdot w_{\text{d}}$, where $\text{d} = \text{cpu}, \text{gpu}, \text{phi}$.

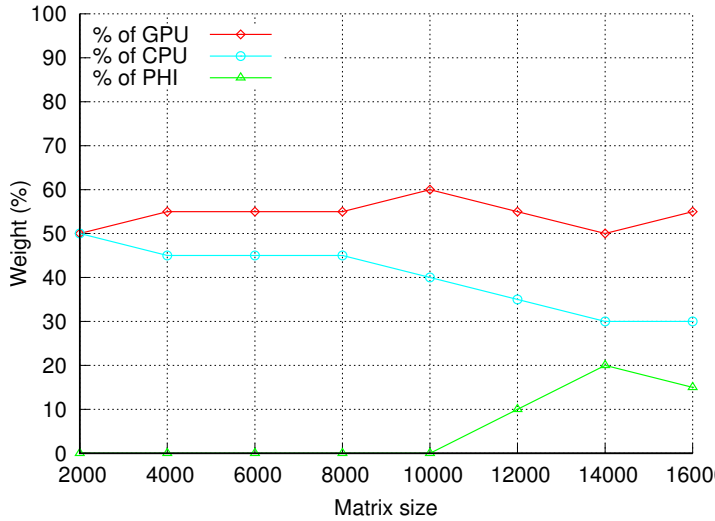


Figure 5.21: Functional model in graphics for the execution time of the matrix multiplication.

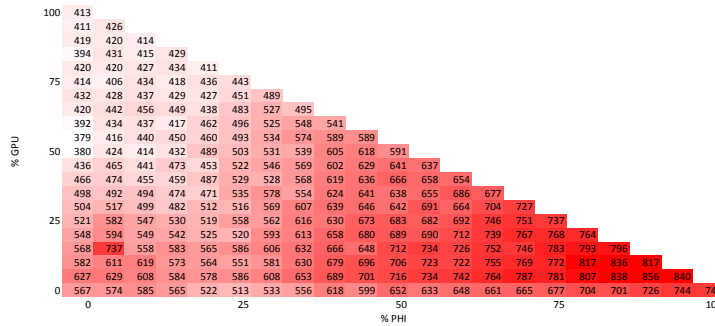


Figure 5.22: Energy consumption in joules for a matrix product of size $n = 8000$ varying the weight of workload on each device.

accelerators are used. This is due, on one hand, to the high energy consumption of the PHI and, on the other hand, to that the NVIDIA GPUs has two different performance states (when idle) that are difficult to control and disturb the actual energy measurement when the device is idle.

5.5.3 Conclusions

This work has presented an application for matrix multiplications in a heterogeneous node composed by a multicore CPU and two very different accelerators. We have shown that it is not difficult to implement the application using OpenMP subsections. However, the incompatibility among compiler versions can make this task a bit cumbersome, and in addition, selecting the exact suitable value for the large number

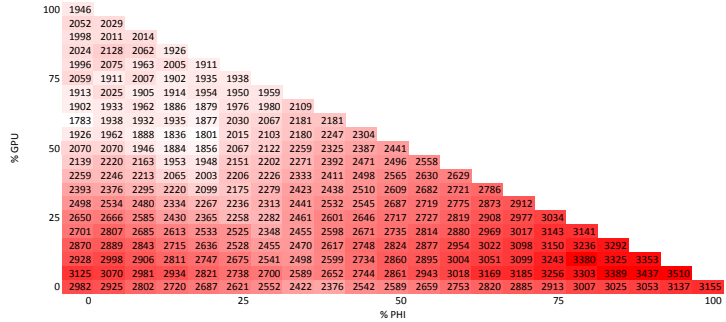


Figure 5.23: Energy consumption in joules for a matrix product of size $n = 14000$ varying the weight of workload on each device.

of environment variables is an arduous task that highly affects the performance of the application.

We have reduced the study of our application to a particular case in which all matrices involved are square. This case is motivated by the existence of many applications that involve only square matrices like, e.g. the evaluation of matrix polynomials, which is the core operation to obtain matrix functions using the Taylor method. However, the study can be extended to rectangular matrices with little effort. We have developed a functional model for the runtime so that we can select the proper amount of work to do by each device. In our node, the K40 is the most speedy device, far more than the Xeon Phi, which only has opportunity to contribute to the computation on matrices larger than $n = 10000$. Furthermore, the Xeon Phi is currently the most expensive device in terms of energy consumption, and the K40 is the most energy efficient. Our study on the energy consumption resulted in a quite simple behaviour, i.e. the lowest total energy consumption is achieved when the GPU is used in a similar proportion as that selected to achieve the lowest execution time, provided the Xeon Phi is not used at all. It was impossible to obtain an accurate measure of dynamic energy due to specific behaviour of the GPU, which changes between two different performance states (when idle) in an unpredictable way. Finally, we proposed a heterogeneous matrix multiplication system to make easy the programmability of algorithms based on a heterogeneous matrix multiplication.

5.6 Summary

Energy is a challenge for very large scale systems. Thus, there are several challenges that arise for midterm research, and possible approaches to address the future research topics and challenges:

Ultra large scale profiling and monitoring for reporting energy usage: Possible approaches include lightweight monitoring approaches using hardware support, accurate models allowing energy estimation (as in the case of RAPL), big data / machine learning techniques to cope with data sizes and complexity.

Multi-objective energy metrics: Metrics relevant for ultrascale, for example power proportionality, scalability per J, trade-off with resilience, efficiency scalability, data movement efficiency on all levels.

Models and simulation of energy consumption of UCS: In general applying simulations to ultrascale systems will require simplified but accurate empirical modeling, stochastic approaches, and machine learning. Research may include machine learning approaches, such as decision trees, linear regression, regression trees and artificial neural network. Energy simulators should include models of energy consumption of heterogeneous hardware to simulate it in ultrascale. Such approaches will work with data collected in heterogeneous cloud infrastructure, running different types of hardware configurations (meaning different CPUs and different types of network hardware and topologies) and different types of virtualization technologies. Energy simulators should also model lightweight virtualization techniques.

Energy efficient resource management and scheduling: Possible approaches may include approximate algorithms taking into account risk of failures and assuming inaccuracy of information. They could use stochastic and machine learning methods for management. On single resources level today technology allows the logic isolation of jobs at a still significant performance costs. Therefore, low cost context switch and low cost process reallocation are features that need to be improved in order to achieve effective resource sharing policies. For virtualized infrastructures the workload dynamic consolidation techniques should take advantage of very lightweight virtualisztion techniques and attempt to allocate large numbers of virtualized tasks in such a way to efficiently use the shared (possibly heterogeneous) resources. Additionally, adjusting execution to energy demand response and local renewables characteristics can help to significantly reduce energy costs and carbon footprint of ultrascale systems.

Energy efficient algorithms and eco-design of applications: Applications should be automatically analyzed and represented as a graph where nodes represent tasks that can be compiled and run in any of the computing elements of the system. Many bibliography addresses the scheduling of such kind of graphs but it is still a challenge to automatically generate quality graphs from application?s code, especially with a focus on maximizing energy efficiency.

To face those challenges, we will need the adoption of intelligent methods for modeling and improving energy efficiency, to increase awareness and focus on energy efficiency in all UCS areas, and to design software taking the advantage of heterogeneous hardware and infrastructure.

Table 5.8 Ideal Manpower repartition to operate a medium-size HPC facility.

SMP Services	**1 *FTE***
Operating Systems	10%
Compilers	10%
COTS Applications	15%
Open Source Applications	30%
Storage	20%
Change Management	10%
Hardware	5%
SAN Storage and Parallel File System	**1 *FTE***
Hardware	30%
Software (GPFS,Lustre,OneFS...)	50%
Allocation and Provisioning	10%
Capacity Planning	10%
Parallel Cluster	**1.6*FTE***
Operating Systems	10%
Compilers	10%
COTS Applications	20%
Open Source Applications	30%
Storage	20%
Change Management	10%
Hardware	5%
Queueing systems	10%
Internal Cluster Network	5%
Monitoring	20%
Database	**1 *FTE***
Installation and Configuration	10%
Tuning and Optimization	20%
Backups and DR	10%
Database Design	40%
Database Administration	20%
Web Services	**40% *FTE***
Operating Systems	10%
Apache	10%
Application Server	10%
Server-based security	10%
Backup / Recovery	**35% *FTE***
Server Admin	20%
Client Admin	10%
Capacity Planning	5%
Scientific Programming	**2 *FTE***
Development	100%
Debugging/Testing	60%
Profiling/Optimization	40%

Table 5.9 `chaos` *and* `gaia` *Computing Nodes Characteristics (as of 2015) and Estimated TCO evaluation per computing node.*

	Node	CPUs/GPU	Memory [GB]	Nodes	CPU Family	GFLOPS	**Hourly Cost ($)**
CHAOS	d-cluster1	12	24	16	westmere (2.26 GHz)	108.48	0.386
	e-cluster1	16	32	16	sandybridge (2.20 GHz)	281.60	0.380
	h-cluster1	12	24	32	westmere (2.26 GHz)	108.48	0.375
	r-cluster1	32	1024	1	nehalem (2.26 GHz)	289.28	1.760
	s-cluster1	16	32	16	sandybridge (2.20 GHz)	281.60	0.380
GAIA	gaia-[1-60]	12	48	60	westmere (2.26 GHz)	108.48	0.400
	gaia-[123-154]	12	48	32	westmere (3.07 GHz)	147.36	0.291
	gaia-[61-62]	12/1792	24	2	westmere (2.26 GHz)	108.48	0.588
	gaia-[63-72]	12/10240	24	10	westmere (2.26 GHz)	108.48	0.545
	gaia-[75-79]	16/12480	64	5	sandybridge (2.20 GHz)	281.60	0.524
	gaia-[83-122]	12	48	40	westmere (2.93 GHz)	140.64	0.291
	gaia-73	160	1024	1	sandybridge (2.00 GHz)	2560.00	2.596
	gaia-74	32	1024	1	sandybridge (2.40 GHz)	614.40	1.463

Table 5.10 EC2 Instance Types – Grouped by Family.

Instance Family	**Instance Type**	**Processor Microarchitecture**	**Introduction Date**
General Purpose	m1	Xeon Family	2006-08-23
	m3	Ivy Bridge-EP	2012-10-31
	t2	Xeon Family	2014-07-01
	m4	Haswell-EP	2015-06-11
	m5	Skylake	2017-11-28
Memory Optimized	m2	Xeon Family	2010-02-22
	cr1	Sandy Bridge-EP	2013-01-21
	r3	Ivy Bridge-EP	2014-04-10
	r4	Broadwell	2016-11-30
Compute Optimized	c1	Xeon Family	2008-08-08
	cc1	Nehalem-EP	2010-07-13
	cc2	Sandy Bridge-EP	2011-11-14
	c3	Ivy Bridge-EP	2013-11-14
	c4	Haswell-EP	2014-11-13
	c5	Skylake	2017-11-06
Storage Optimized	hi1	Xeon Family	2012-07-18
	hs1	Sandy Bridge-EP	2012-12-21
	i2	Ivy Bridge-EP	2013-12-20
	i3	Broadwell	2017-02-23
Dense Storage	d2	Haswell-EP	2015-03-30
GPU	cg1	Nehalem-EP	2010-11-14
	g2	Sandy Bridge-EP	2013-11-05
	g3	Broadwell	2017-07-13
Micro	t1	Xeon Family	2009-10-26
	t2	Haswell	2014-07-01

Table 5.11 Coefficients of the New EC2 Price Model for On Demand Instances, with Fitting Performance Evaluation.

Generation	GFLOPS (α)	MemGiB (β)	DiskGiB (γ)	GPUs (δ)	Adj. R^2	P-Value
1st (m1, c1, m2, cg1)	0.0039522	0.0061130	0.0000670	0.0015395	0.9999909	0e+00
2nd (cc2, m3, hi1)	-0.0035266	0.0355353	0.0007284	0.0000000	0.9999785	1e-07
3rd (hs1, cr1, g2, c3)	0.0017209	0.0106101	0.0000655	0.0001644	1.0000000	0e+00
4th (i2, r3, c4)	0.0009952	0.0081883	0.0007605	0.0000000	0.9998832	0e+00
5th (m4, d2)	0.0000000	0.0173750	0.0000342	0.0000000	1.0000000	0e+00

Hardware	Model	Cores	RAM (GB)
Nova: Dell PowerEdge R430	Intel Xeon E5-2620 (2.1GHz)	2x8	64
Taurus: Dell PowerEdge R720	Intel Xeon E5-2630 (2.3GHz)	2x6	32
Sagittaire: Sun Fire V20z	AMD Opteron 250 (2.4GHz)	2	2

Table 5.12 Servers characteristics of the Grid'5000 Lyon site

Table 5.13 Meaning of shell variables used to execute the heterogeneous matrix multiplication application.

Variable name	Meaning
`OMP_NESTED`:	Set to `TRUE` to ensure that MKL uses more than one thread when called inside an OpenMP subsection.
`MKL_NUM_THREADS`:	Number of threads used by MKL (CPU).
`MKL_DYNAMIC`:	Set to `FALSE` to avoid MKL automatically selects the number of threads (CPU).
`MKL_MIC_ENABLE`:	Set to 0 to avoid the Xeon Phi is used to accelerate the CPU computation.
`MIC_ENV_PREFIX`:	Specifies the environment variables with prefix `MIC` will address only the PHI.
`MIC_OMP_NUM_THREADS`:	Number of threads used by the PHI to execute MKL routines.
`MIC_KMP_AFFINITY`, `MIC_USE_2MB_BUFFERS`:	These variables control the efficiency of the Xeon Phi in the execution of the matrix multiplication routine. They have been set to such values according to the advice of Intel documentation.

	$n \in [2000, 10000]$	$n \in [10000, 14000]$
$w_{\text{phi}} =$	0	$\frac{n}{200} - 50$
$w_{\text{gpu}} =$	$\frac{n}{800} + 47.5$	$-\frac{n}{400} + 85$
$w_{\text{cpu}} =$	$100 - w_{\text{gpu}}$	$100 - (w_{\text{phi}} + w_{\text{gpu}})$

Table 5.14 Functions of the weight for each device for the execution time of the matrix multiplication.

References

References

[1] Janetschek M, Prodan R, Benedict S. A Workflow Runtime Environment for Manycore Parallel Architectures. Future Generation Computer Systems. 2017;75:330–347.

[2] van der Linde A, Fouto P, Leitão J, et al. Legion: Enriching Internet Services with Peer-to-Peer Interactions. In: Proceedings of the 26th International Conference on World Wide Web. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee; 2017. p. 283–292. Available from: https://doi.org/10.1145/3038912.3052673.

[3] Bagein M, Barbosa J, Blanco V, et al. Energy Efficiency for Ultrascale Systems: Challenges and Trends from Nesus Project. Intl J on Supercomputing Frontiers and Innovations. 2015;2(2):105–131. See http://superfri.org/superfri/article/view/48.

[4] RTE. eCO2mix: energy mix for RTE, the French transmission system operator; 2018. https://www.rte-france.com/en/eco2mix/eco2mix-mix-energetique-en.

[5] ?Memorandum of Understanding for the implementation of a European Concerted Research Action designated as COST Action IC1305: Network for Sustainable Ultrascale Computing (NESUS)?; 2014. Available from: https://e-services.cost.eu/files/domain_files/ICT/Action_IC1305/mou/IC1305-e.pdf,November2013.

[6] Sousa L, Kropf P, Kuonene P, et al. A Roadmap for Research in Sustainable Ultrascale Systems. University Carlos III of Madrid; 2017. 0. Available from: https://www.irit.fr/~Georges.Da-Costa/NESUS-research_roadmap.pdf.

[7] Da Costa G, Fahringer T, Gallego JAR, et al. Exascale machines require new programming paradigms and runtimes. Supercomputing frontiers and innovations. 2015;2(2):6–27.

[8] Fortune S, Wyllie J. Parallelism in Random Access Machines. In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing. STOC '78. New York, NY, USA: ACM; 1978. p. 114–118. Available from: http://doi.acm.org/10.1145/800133.804339.

[9] Valiant LG. A Bridging Model for Parallel Computation. Commun ACM. 1990 Aug;33(8):103–111. Available from: http://doi.acm.org/10.1145/79173.79181.

[10] Culler D, Karp R, Patterson D, et al. LogP: towards a realistic model of parallel computation. In: Proceedings of the fourth ACM SIGPLAN

symposium on Principles and practice of parallel programming. PPOPP '93. New York, NY, USA: ACM; 1993. p. 1–12.

[11] Gautier T, Besseron X, Pigeon L. KAAPI: A Thread Scheduling Runtime System for Data Flow Computations on Cluster of Multi-processors. In: Proceedings of the 2007 International Workshop on Parallel Symbolic Computation. PASCO '07. ACM; 2007. p. 15–23.

[12] Augonnet C, Thibault S, Namyst R, et al. StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. Concurr Comput : Pract Exper. 2011 Feb;23(2):187–198.

[13] Bosilca G, Bouteiller A, Danalis A, et al. DAGuE: A Generic Distributed DAG Engine for High Performance Computing. In: 2011 IEEE IPDPSW; 2011. p. 1151–1158.

[14] Bui TN, Jones C. A heuristic for reducing fill-in in sparse matrix factorization. In: Proceedings of the 6th SIAM Conference on Parallel Processing for Scientific Computing. SIAM; 1993. .

[15] Hendrickson B, Leland R. A Multilevel Algorithm for Partitioning Graphs. In: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing. Supercomputing '95. ACM; 1995. Available from: http://doi.acm.org/10.1145/224170.224228.

[16] Catalyurek U, Aykanat C. Decomposing Irregularly Sparse Matrices for Parallel Matrix-Vector Multiplication. In: Proceedings of the Third International Workshop on Parallel Algorithms for Irregularly Structured Problems. IRREGULAR '96. Springer-Verlag; 1996. p. 75–86.

[17] Hendrickson B, Kolda TG. Partitioning Rectangular and Structurally Unsymmetric Sparse Matrices for Parallel Processing. SIAM Journal on Scientific Computing. 2000;21(6):2048–2072.

[18] Cierniak M, Zaki MJ, Li W. Compile-time scheduling algorithms for a heterogeneous network of workstations. The Computer Journal. 1997;40(6):356–372.

[19] Beaumont O, Boudet V, Rastello F, et al. Matrix multiplication on heterogeneous platforms. Parallel and Distributed Systems, IEEE Transactions on. 2001;12(10):1033–1051.

[20] Kalinov A, Lastovetsky A. Heterogeneous Distribution of Computations Solving Linear Algebra Problems on Networks of Heterogeneous Computers. Journal of Parallel and Distributed Computing. 2001;61:520–535.

[21] Lastovetsky AL, Reddy R. Data partitioning with a realistic performance model of networks of heterogeneous computers. In: Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International. IEEE; 2004. p. 104.

[22] Lastovetsky A, Twamley J. Towards a realistic performance model for networks of heterogeneous computers. In: High Performance Computational Science and Engineering. Springer; 2005. p. 39–57.

[23] Lastovetsky A, Reddy R. Data partitioning with a functional performance model of heterogeneous processors. International Journal of High Performance Computing Applications. 2007;21(1):76–90.

[24] Lastovetsky A, Reddy R. Data Distribution for Dense Factorization on Computers with Memory Heterogeneity. Parallel Computing. 2007 Dec;33(12).

[25] Lastovetsky A, Szustak L, Wyrzykowski R. Model-based optimization of EULAG kernel on Intel Xeon Phi through load imbalancing. IEEE Transactions on Parallel and Distributed Systems. 2017;28(3):787–797.

[26] Lastovetsky A, Reddy R. New Model-Based Methods and Algorithms for Performance and Energy Optimization of Data Parallel Applications on Homogeneous Multicore Clusters. IEEE Transactions on Parallel and Distributed Systems. 2017;28(4):1119–1133.

[27] Alexandrov A, Ionescu MF, Schauser KE, et al. LogGP: incorporating long messages into the LogP model - one step closer towards a realistic model for parallel computation. In: Proc. of the seventh annual ACM symposium on Parallel algorithms and architectures. SPAA '95. NY, USA; 1995. p. 95–105.

[28] Kielmann T, Bal HE, Verstoep K. Fast Measurement of LogP Parameters for Message Passing Platforms. In: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing. IPDPS '00. London, UK, UK: Springer-Verlag; 2000. p. 1176–1183.

[29] Bosque JL, Perez LP. HLogGP: a new parallel computational model for heterogeneous clusters. In: Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on; 2004. p. 403–410.

[30] Lastovetsky A, Mkwawa IH, O'Flynn M. An accurate communication model of a heterogeneous cluster based on a switch-enabled Ethernet network. In: Parallel and Distributed Systems, 2006. ICPADS 2006. 12th International Conference on. vol. 2; 2006. p. 6 pp.–.

[31] Cameron KW, Ge R, Sun XH. $\log_m P$ and $\log_3 P$: Accurate Analytical Models of Point-to-Point Communication in Distributed Systems. Computers IEEE Transactions on. 2007;56(3):314–327.

[32] Rico-Gallego JA, Díaz-Martín JC, Lastovetsky AL. Extending τ–Lop to model concurrent MPI communications in multicore clusters. Future Generation Computer Systems. 2016;61:66 – 82.

[33] Rico-Gallego JA, Lastovetsky AL, Díaz-Martín JC. Model-Based Estimation of the Communication Cost of Hybrid Data-Parallel Applications on Heterogeneous Clusters. IEEE Transactions on Parallel and Distributed Systems. 2017 Nov;28(11):3215–3228.

[34] Clarke D, Zhong Z, Rychkov V, et al. FuPerMod: a software tool for the optimization of data-parallel applications on heterogeneous platforms. The Journal of Supercomputing. 2014;69:61– 69.

[35] Beaumont O, Boudet V, Rastello F, et al. Matrix Multiplication on Heterogeneous Platforms. IEEE Trans Parallel Distrib Syst. 2001 Oct;12(10):1033–1051. Available from: http://dx.doi.org/10.1109/71.963416.

[36] Malik T, Rychkov V, Lastovetsky A. Network-aware optimization of communications for parallel matrix multiplication on hierarchical HPC platforms. Concurrency and Computation: Practice and Experience. 2016 03/2016;28:802–821.

[37] Bellosa F. The benefits of event: driven energy accounting in power-sensitive systems. In: Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system. ACM; 2000. .

[38] Isci C, Martonosi M. Runtime power monitoring in high-end processors: Methodology and empirical data. In: 36th annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society; 2003. p. 93.

[39] Economou D, Rivoire S, Kozyrakis C, et al. Full-system power analysis and modeling for server environments. In: In Proceedings of Workshop on Modeling, Benchmarking, and Simulation; 2006. p. 70–77.

[40] Basmadjian R, Ali N, Niedermeier F, et al. A methodology to predict the power consumption of servers in data centres. In: 2nd International Conference on Energy-Efficient Computing and Networking. ACM; 2011. .

[41] Bircher WL, John LK. Complete System Power Estimation Using Processor Performance Events. IEEE Transactions on Computers. 2012 Apr;61(4):563–577.

[42] Hong H Sunpyand Kim. An Integrated GPU Power and Performance Model. SIGARCH Comput Archit News. 2010 Jun;38(3):280–289.

[43] Song S, Su C, Rountree B, et al. A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures. In: 27th IEEE International Parallel & Distributed Processing Symposium (IPDPS). IEEE Computer Society; 2013. p. 673–686.

[44] CUPTI. CUDA Profiling Tools Interface; 2018. Available from: https://developer.nvidia.com/cuda-profiling-tools-interface.

[45] Wang H, Cao Y. Predicting power consumption of GPUs with fuzzy wavelet neural networks. Parallel Computing. 2015 May;44:18–36.

[46] Top500. Top 500. The List - November 2017; 2018. Available from: https://www.top500.org/lists/2017/11/.

[47] Shao YS, Brooks D. Energy Characterization and Instruction-level Energy Model of Intel's Xeon Phi Processor. In: Proceedings of the 2013 International Symposium on Low Power Electronics and Design. ISLPED '13. IEEE Press; 2013. .

[48] Ou J, Prasanna VK. Rapid energy estimation of computations on FPGA based soft processors. In: SOC Conference, 2004. Proceedings. IEEE International; 2004. .

[49] Wang X, Ziavras SG, Hu J. System-Level Energy Modeling for Heterogeneous Reconfigurable Chip Multiprocessors. In: 2006 International Conference on Computer Design; 2006. .

[50] Al-Khatib Z, Abdi S. Operand-Value-Based Modeling of Dynamic Energy Consumption of Soft Processors in FPGA. In: International Symposium on Applied Reconfigurable Computing. Springer; 2015. p. 65–76.

[51] Lively C, Wu X, Taylor V, et al. Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems. Computer Science-Research and Development. 2012;27(4):245–253.

[52] PAPI. Performance Application Programming Interface 5.6.0; 2018. Available from: http://icl.cs.utk.edu/papi/.

[53] Bosilca G, Ltaief H, Dongarra J. Power profiling of Cholesky and QR factorizations on distributed memory systems. Computer Science-Research and Development. 2014;29(2):139–147.

[54] Witkowski M, Oleksiak A, Piontek T, et al. Practical Power Consumption Estimation for Real Life HPC Applications. Future Gener Comput Syst. 2013 Jan;29(1).

[55] Jarus M, Oleksiak A, Piontek T, et al. Runtime power usage estimation of HPC servers for various classes of real-life applications. Future Generation Computer Systems. 2014;36.

[56] Lastovetsky A, Manumachu RR. New Model-Based Methods and Algorithms for Performance and Energy Optimization of Data Parallel Applications on Homogeneous Multicore Clusters. IEEE Transactions on Parallel and Distributed Systems. 2017;28(4):1119–1133.

[57] McCullough JC, Agarwal Y, Chandrashekar J, et al. Evaluating the Effectiveness of Model-based Power Characterization. In: Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference. USENIXATC'11. USENIX Association; 2011. .

[58] Hackenberg D, Ilsche T, Schöne R, et al. Power measurement techniques on standard compute nodes: A quantitative comparison. In: Performance analysis of systems and software (ISPASS), 2013 IEEE international symposium on. IEEE; 2013. p. 194–204.

[59] Rotem E, Naveh A, Ananthakrishnan A, et al. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. IEEE Micro. 2012 March;32(2):20–27.

[60] O'Brien K, Pietri I, Reddy R, et al. A Survey of Power and Energy Predictive Models in HPC Systems and Applications. ACM Computing Surveys. 2017;50(3). Available from: http://doi.org/10.1145/3078811.

[61] Shahid A, Fahad M, Reddy R, et al. Additivity: A Selection Criterion for Performance Events for Reliable Energy Predictive Modeling. Supercomputing Frontiers and Innovations. 2017;4(4).

[62] Treibig J, Hager G, Wellein G. Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In: Parallel Processing Workshops (ICPPW), 2010 39th International Conference on. IEEE; 2010. p. 207–216.

[63] Mobius C, Dargie W, Schill A. Power Consumption Estimation Models for Processors, Virtual Machines, and Servers. IEEE Transactions on Parallel and Distributed Systems. 2014;25(6).

[64] Inacio EC, Dantas MAR. A Survey into Performance and Energy Efficiency in HPC, Cloud and Big Data Environments. Int J Netw Virtual Organ. 2014 Mar;14(4).

[65] Tan L, Kothapalli S, Chen L, et al. A survey of power and energy efficient techniques for high performance numerical linear algebra operations. Parallel Computing. 2014 Dec;40.

[66] Dayarathna M, Wen Y, Fan R. Data Center Energy Consumption Modeling: A Survey. IEEE Communications Surveys & Tutorials. 2016;18(1):732–794.

[67] Mezmaz M, Melab N, Kessaci Y, et al. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. Journal of Parallel and Distributed Computing. 2011;71(11):1497 – 1508.

[68] Fard HM, Prodan R, Barrionuevo JJD, et al. A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments. In: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012). CCGRID '12. IEEE Computer Society; 2012. p. 300–309.

[69] Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Generation Computer Systems. 2012;28(5):755 – 768. Special Section: Energy efficiency in large-scale distributed systems.

[70] Kessaci Y, Melab N, Talbi EG. A Pareto-based Metaheuristic for Scheduling HPC Applications on a Geographically Distributed Cloud Federation. Cluster Computing. 2013 Sep;16(3):451–468.

[71] Durillo JJ, Nae V, Prodan R. Multi-objective energy-efficient workflow scheduling using list-based heuristics. Future Generation Computer Systems. 2014;36:221 – 236.

[72] Freeh VW, Lowenthal DK, Pan F, et al. Analyzing the Energy-Time Trade-Off in High-Performance Computing Applications. IEEE Trans Parallel Distrib Syst. 2007 Jun;18(6).

[73] Ahmad I, Ranka S, Khan SU. Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy. In: Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on; 2008. p. 1–6.

[74] Balaprakash P, Tiwari A, Wild SM. In: Jarvis AS, Wright AS, Hammond DS, editors. Multi Objective Optimization of HPC Kernels for Performance, Power, and Energy. Springer International Publishing; 2014. p. 239–260.

[75] Drozdowski M, Marszalkowski JM, Marszalkowski J. Energy trade-offs analysis using equal-energy maps. Future Generation Computer Systems. 2014;36:311–321.

[76] Marszalkowski JM, Drozdowski M, Marszalkowski J. Time and Energy Performance of Parallel Systems with Hierarchical Memory. Journal of Grid Computing. 2016;14(1):153–170.

[77] Reddy R, Lastovetsky A. Bi-Objective Optimization of Data-Parallel Applications on Homogeneous Multicore Clusters for Performance and Energy. IEEE Transactions on Computers. 2018;64(2):160–177.

[78] Juve G, Chervenak A, Deelman E, et al. Characterizing and profiling scientific workflows. Future Generation Computer Systems. 2013;29(3):682–692.

[79] Fahringer T, Prodan R, Duan R, et al. ASKALON: A Development and Grid Computing Environment for Scientific Workflows. In: Taylor IJ, Deelman E, Gannon DB, et al., editors. Workflows for e-Science. Springer; 2007. p. 450–471.

[80] Altintas I, Berkley C, Jaeger E, et al. Kepler: an extensible system for design and execution of scientific workflows. In: Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.; 2004. p. 423–424.

[81] Tristan Glatard DLXP Johan Montagnat. Flexible and Efficient Workflow Deployment of Data-Intensive Applications On Grids With MOTEUR. The International Journal of High Performance Computing Applications. 2008;22(3):347–360.

[82] Taylor I, Shields M, Wang I, et al. Triana Applications within Grid Computing and Peer to Peer Environments. Journal of Grid Computing. 2003 Jun;1(2):199–217.

[83] Kacsuk P. PâĂŘ-GRADE portal family for grid infrastructures. Concurrency and Computation: Practice and Experience. 2011;23(3):235–245.

[84] Deelman E, Vahi K, Juve G, et al. Pegasus, a workflow management system for science automation. Future Generation Computer Systems. 2015;46:17–35.

[85] Durillo JJ, Prodan R, Barbosa JG. Pareto tradeoff scheduling of workflows on federated commercial clouds. Simulation Modelling Practice and Theory. 2015;58:95–111.

[86] Arabnejad H, Barbosa JG. Budget constrained scheduling strategies for online workflow applications. In: International Conference on Computational Science and Its Applications. Springer; 2014. p. 532–545.

[87] Ullman JD. NP-complete scheduling problems. Journal of Computer and System sciences. 1975;10(3):384–393.

[88] Topcuoglu H, Hariri S, Wu MY. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems. 2002 3;13(3):260–274.

[89] Wieczorek M, Hoheisel A, Prodan R. Towards a General Model of the Multi-Criteria Workflow Scheduling on the Grid. Future Generations Computer Systems. 2009;25(3):237–256.

[90] Maheswaran M, Ali S, Siegel HJ, et al. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of parallel and distributed computing. 1999;59(2):107–131.

[91] Arabnejad H, Barbosa JG. List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Transactions on Parallel and Distributed Systems. 2014;25(3):682–694.

[92] Bittencourt LF, Sakellariou R, Madeira ER. Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm. In: Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on. IEEE; 2010. p. 27–34.

[93] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. Communications of the ACM. 2010;53(4):50–58.

[94] Leitão J, Pereira J, Rodrigues L. Epidemic Broadcast Trees. In: Proceedings of SRDS.2007; 2007. p. 301–310.

[95] Leitão J, Pereira J, Rodrigues L. HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast. In: Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on; 2007. p. 419–429.

[96] Shapiro M, Preguiça N, Baquero C, et al. Conflict-free replicated data types. INRIA; 2011. RR-7687.

[97] Almeida PS, Shoker A, Baquero C. Efficient state-based crdts by delta-mutation. In: International Conference on Networked Systems. Springer; 2015. p. 62–76.

[98] Carlos Baquero PSA, Shoker A. Making Operation-Based CRDTs Operation-Based. In: Distributed Applications and Interoperable Systems - 14th IFIP WG 6.1 International Conference, DAIS 2014, Held as Part of the 9th International Federated Conference on Distributed Computing Techniques, DisCoTec 2014, Berlin, Germany, June 3-5, 2014, Proceedings; 2014. p. 126–140. Available from: http://dx.doi.org/10.1007/978-3-662-43352-2_11.

[99] Bonomi F, Milito R, Zhu J, et al. Fog Computing and Its Role in the Internet of Things. Proceedings of the first edition of the MCC workshop on Mobile cloud computing. 2012;p. 13–16. Available from: http://doi.acm.org/10.1145/2342509.2342513$\delimiter"026E30F$npapers2://publication/doi/10.1145/2342509.2342513.

[100] Yi S, Li C, Li Q. A Survey of Fog Computing: Concepts, Applications and Issues. In: Proceedings of the 2015 Workshop on Mobile Big Data. Mobidata '15. New York, NY, USA: ACM; 2015. p. 37–42. Available from: http://doi.acm.org/10.1145/2757384.2757397.

[101] Verbelen T, Simoens P, De Turck F, et al. Cloudlets: Bringing the cloud to the mobile user. In: Proceedings of the third ACM workshop on Mobile cloud computing and services. ACM; 2012. p. 29–36.

[102] Fernando N, Loke SW, Rahayu W. Mobile cloud computing: A survey. Future generation computer systems. 2013;29(1):84–106.

[103] Hu YC, Patel M, Sabella D, et al. Mobile edge computingâĂŤA key technology towards 5G. ETSI white paper. 2015;11(11):1–16.

[104] Cisco. Cisco IOx Data Sheet; 2016. Available from: http://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/iox/datasheet-c78-736767.html.

[105] Dell. Dell Edge Gateway 5000; 2016. Available from: http://www.dell.com/us/business/p/dell-edge-gateway-5000/pd?oc=xctoi5000us.

[106] Milojicic DS, Kalogeraki V, Lukose R, et al. Peer-to-peer computing. 2002;.

[107] Jelasity M, Montresor A, Babaoglu O. Gossip-based aggregation in large dynamic networks. ACM Transactions on Computer Systems (TOCS). 2005;23(3):219–252.

[108] Akyildiz IF, Su W, Sankarasubramaniam Y, et al. Wireless sensor networks: a survey. Computer networks. 2002;38(4):393–422.

[109] Gilbert S, Lynch N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services. SIGACT News. 2002 Jun;33(2):51–59.

[110] Meiklejohn C, Van Roy P. Lasp: A Language for Distributed, Coordination-Free Programming. In: Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming (PPDP 2015). ACM; 2015. p. 184–195.

[111] Carvalho N, Pereira J, Oliveira R, et al. Emergent Structure in Unstructured Epidemic Multicast. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07). Edinburgh, Scotland, UK; 2007. p. 481 – 490.

[112] Balegas V, Serra D, Duarte S, et al. Extending Eventually Consistent Cloud Databases for Enforcing Numeric Invariants. In: Proceedings of SRDS 2015. Montréal, Canada: IEEE Computer Society; 2015. p. 31–36. Available from: http://lip6.fr/Marc.Shapiro/papers/numeric-invariants-SRDS-2015.pdf.

[113] Najafzadeh M, Shapiro M, Balegas V, et al. Improving the scalability of geo-replication with reservations. In: ACM SIGCOMM - Distributed Cloud Computing (DCC). Dresden, Germany; 2013. Available from: http://lip6.fr/Marc.Shapiro//papers/escrow-DCC-2013.pdf.

[114] Gotsman A, Yang H, Ferreira C, et al. 'Cause I'M Strong Enough: Reasoning About Consistency Choices in Distributed Systems. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York, NY, USA: ACM; 2016. p. 371–384. Available from: http://doi.acm.org/10.1145/2837614.2837625.

[115] Akkoorath DD, Tomsic A, Bravo M, et al. Cure: Strong Semantics Meets High Availability and Low Latency. INRIA; 2016. RR-8858.

[116] Lasp: The Missing part of Erlang distribution;. Accessed: 2018-04-27. http://www.lasp-lang.org.

[117] Meiklejohn C, Enes V, Yoo J, et al. Practical Evaluation of the Lasp Programming Model at Large Scale. In: Proceedings of the 19th International Symposium on Principles and Practice of Declarative Programming (PPDP 2017). ACM; 2017. p. 109–114.

[118] Bichot CE, Siarry P. Graph partitioning. John Wiley & Sons; 2013.

[119] Shewchuk JR. Allow Me to Introduce Spectral and Isoperimetric Graph Partitioning; 2016. Available from: http://www.cs.berkeley.edu/~jrs/papers/partnotes.pdf.

[120] Bellman R. Introduction to matrix analysis. vol. 960. SIAM;.

[121] Chung FR. Laplacians of graphs and Cheeger's inequalities. Combinatorics, Paul Erdos is Eighty. 1996;2(157-172):13–2.

[122] Spielman DA, Teng SH. Spectral partitioning works: Planar graphs and finite element meshes. Linear Algebra and its Applications. 2007;421(2):284–305.

[123] Gantmakher FR. The theory of matrices. vol. 131. American Mathematical Soc.; 1998.

[124] Berman A, Plemmons RJ. Nonnegative matrices. vol. 9. SIAM; 1979.

[125] Fiedler M. Algebraic connectivity of graphs. Czechoslovak mathematical journal. 1973;23(2):298–305.

[126] Mohar B. Isoperimetric numbers of graphs. Journal of Combinatorial Theory, Series B. 1989;47(3):274–291.

[127] Van Driessche R, Roose D. An improved spectral bisection algorithm and its application to dynamic load balancing. Parallel computing. 1995;21(1):29–48.

[128] Hendrickson B, Leland R. An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM Journal on Scientific Computing. 1995;16(2):452–469.

[129] Lancaster P, Tismenetsky M. The theory of matrices: with applications. Elsevier; 1985.

[130] Chevalier C, Pellegrini F. PT-Scotch: A tool for efficient parallel graph ordering. Parallel computing. 2008;34(6):318–331.

[131] Anderson E, Bai Z, Bischof C, et al. LAPACK Users' Guide (Software, Environments and Tools) 3rd Edition;.

[132] Bergamaschi L, Bozzo E. Computing the smallest eigenpairs of the graph Laplacian. SeMA Journal. 2018;75(1):1–16.

[133] Soper AJ, Walshaw C, Cross M. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. Journal of Global Optimization. 2004;29(2):225–241.

[134] Zheng A, Labrinidis A, Pisciuneri PH, et al. PARAGON: Parallel Architecture-Aware Graph Partition Refinement Algorithm. In: EDBT; 2016. p. 365–376.

[135] Fiduccia CM, Mattheyses RM. A linear-time heuristic for improving network partitions. In: Papers on Twenty-five years of electronic design automation. ACM; 1988. p. 241–247.

[136] Wasim MU, Ibrahim AAZA, Bouvry P, et al. Law as a service (LaaS): Enabling legal protection over a blockchain network. In: 2017 14th International Conference on Smart Cities: Improving Quality of Life Using ICT IoT (HONET-ICT); 2017. p. 110–114.

[137] Siewiorek DP, Swarz RS. Reliable Computer Systems (3rd Ed.): Design and Evaluation. Natick, MA, USA: A. K. Peters, Ltd.; 1998.

[138] Snir M, Wisniewski RW, Abraham JA, et al. Addressing failures in exascale computing. IJHPCA. 2014;28(2):129–173. Available from: http://dx.doi.org/10.1177/1094342014522573.

[139] Cappello F, Geist A, Gropp B, et al. Toward Exascale Resilience. IJHPCA. 2009;23(4):374–388. Available from: http://dx.doi.org/10.1177/1094342009347767.

[140] Cappello F. Fault Tolerance in Petascale/ Exascale Systems: Current Knowledge, Challenges and Research Opportunities. IJHPCA. 2009;23(3):212–226. Available from: http://dx.doi.org/10.1177/1094342009106189.

[141] Avizienis A, Laprie JC, Randell B, et al. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing. 2004;1:11–33.

[142] Elnozahy ENM, Alvisi L, Wang YM, et al. A Survey of Rollback-recovery Protocols in Message-passing Systems. ACM Comput Surv. 2002 Sep;34(3):375–408. Available from: http://doi.acm.org/10.1145/568522.568525.

[143] Chen Z, Fagg GE, Gabriel E, et al. Fault Tolerant High Performance Computing by a Coding Approach. In: Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. PPoPP '05. New York, NY, USA: ACM; 2005. p. 213–223. Available from: http://doi.acm.org/10.1145/1065944.1065973.

[144] Vosoughi A, Bilal K, Khan SU, et al. A multidimensional robust greedy algorithm for resource path finding in large-scale distributed networks. In: Proceedings of the 8th International Conference on Frontiers of Information Technology. FIT '10. New York, NY, USA: ACM; 2010. p. 16:1–16:6. Available from: http://doi.acm.org/10.1145/1943628.1943644.

[145] Dumas JG, Roch JL, Tannier E, et al. Foundations of Coding: Compression, Encryption, Error-Correction. Wiley & Sons; 2015. 376 pages.

[146] Mazumder P. Design of a Fault-Tolerant DRAM with New On-Chip ECC. In: Koren I, editor. Defect and Fault Tolerance in VLSI Systems. Springer US; 1989. p. 85–92. Available from: http://dx.doi.org/10.1007/978-1-4615-6799-8_8.

[147] Choi M, Park NJ, George KM, et al. Fault tolerant memory design for HW/SW co-reliability in massively parallel computing systems. In: 2nd Intl. Symp. on Network Computing and Applications, 2003. NCA 2003; 2003. p. 341–348.

[148] Ernst D, Das S, Lee S, et al. Razor: circuit-level correction of timing errors for low-power operation. Micro, IEEE. 2004 Nov;24(6):10–20.

[149] Benini L, De Michelli G. Networks on chips : technology and tools. The Morgan Kaufmann series in systems on silicon. Amsterdam, Boston, Paris: Elsevier Morgan Kaufmann Publishers; 2006. Available from: http://opac.inria.fr/record=b1123186.

[150] Radetzki M, Feng C, Zhao X, et al. Methods for Fault Tolerance in Networks-on-chip. ACM Comput Surv. 2013 Jul;46(1):8:1–8:38. Available from: http://doi.acm.org/10.1145/2522968.2522976.

[151] Park D, Nicopoulos C, Kim J, et al. Exploring Fault-Tolerant Network-on-Chip Architectures. In: Dependable Systems and Networks, 2006. DSN 2006. International Conference on; 2006. p. 93–104.

[152] Muszyński J, Varrette S, Bouvry P. Reducing Efficiency of Connectivity-Splitting Attack on Newscast via Limited Gossip. In: Proc. of the 19th European Event on Bio-Inspired Computation, EvoCOMNET 2016. LNCS. Porto, Portugal: Springer Verlag; 2016. .

[153] Jelasity M, Voulgaris S, Guerraoui R, et al. Gossip-based Peer Sampling. ACM Trans Comput Syst. 2007;25(3). Available from: http://doi.acm.org/10.1145/1275517.1275520.

[154] MPI: A Message-Passing Interface Standard, Version 3.1. MPI forum; 2015. [online] see http://mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf.

[155] Gropp W, Lusk E. Fault Tolerance in Message Passing Interface Programs. The International Journal of High Performance Computing Applications. 2004;18(3):363–372. Available from: https://doi.org/10.1177/1094342004046045.

[156] Yaga D, Mell P, Roby N, et al. Draft NISTIR 8202: Blockchain Technology Overview. **NIST!**; 2018. https://csrc.nist.gov/publications/detail/nistir/8202/draft.

[157] Dumas JG, Lafourcade P, Tichit A, et al. Les blockchains en 50 questions: comprendre le fonctionnement et les enjeux de cette technologie innovante. 1st ed. Collection Sciences Sup. Dunod; 2018. (french).

[158] Wasim MU, Ibrahim A, Bouvry P, et al. Self-Regulated Multi-criteria Decision Analysis: An Autonomous Brokerage-Based Approach for Service Provider Ranking in the Cloud. In: Proc. of the 8th IEEE Intl. conf. on Cloud Computing Technology and Science (CloudCom 2017). Hong Kong; 2017. p. 33–40.

[159] Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. IEEE Access. 2016;4:2292–2303.

[160] Savelyev A. Contract law 2.0:âĂŸSmartâĂŹcontracts as the beginning of the end of classic contract law. Information & Communications Technology Law. 2017;26(2):116–134.

[161] Verbeek M. A guide to modern econometrics. John Wiley & Sons; 2008.

[162] Rummel R. Applied Factor Analysis Northwestern University Press Evanston. IL Google Scholar. 1970;.

[163] Rencher AC. Methods of multivariate analysis. vol. 492. John Wiley & Sons; 2003.

[164] Taylor HM, Karlin S. An introduction to stochastic modeling. Academic press; 2014.

[165] Cooper BF, Silberstein A, Tam E, et al. Benchmarking Cloud Serving Systems with YCSB. In: SoCC' 10 ACM; 2010. .

[166] Papagiannis A, Saloustros G, González-Férez P, et al. Tucana: Design and Implementation of a Fast and Efficient Scale-up Key-value Store. In: Proceedings of the 2016 USENIX Annual Technical Conference (USENIX ATC 16); 2016. p. 537–550.

[167] Apache. HBase;. Accessed: July 13, 2018. https://hbase.apache.org/.

[168] O'Neil P, Cheng E, Gawlick D, et al. The log-structured merge-tree (LSM-tree). Acta Informatica. 1996;33(4):351–385.

[169] Brodal GS, Fagerberg R. Lower Bounds for External Memory Dictionaries. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '03. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics; 2003. p. 546–554. Available from: http://dl.acm.org/citation.cfm?id=644108.644201.

[170] Cooper BF, Silberstein A, Tam E, et al. Benchmarking Cloud Serving Systems with YCSB. In: Proceedings of the 1st ACM Symposium on Cloud Computing. SoCC '10. New York, NY, USA: ACM; 2010. p. 143–154. Available from: http://doi.acm.org/10.1145/1807128.1807152.

[171] Oliker L, Biswas R, Van der Wijngaart R, et al. Performance evaluation and modeling of ultra-scale systems. Parallel Processing for Scientific Computing (Siam 2006). 2006;p. 77–93.

[172] Da Costa G, Fahringer T, Rico-Gallego JA, et al. Exascale machines require new programming paradigms and runtimes. Supercomputing Frontiers and Innovations. 2015 September;2(2):6–27.

[173] Marozzo F, Rodrigo Duro F, Garcia Blas J, et al. A Data-aware Scheduling Strategy for Workflow Execution in Clouds. Concurrency and Computation: Practice and Experience. 2017 December;29(24).

[174] Marozzo F, Talia D, Trunfio P. A Workflow Management System for Scalable Data Mining on Clouds. IEEE Transactions on Services Computing. 2016;PP(99):1–1.

[175] Duro FR, Blas JG, Carretero J. A Hierarchical Parallel Storage System Based on Distributed Memory for Large Scale Systems. In: Proceedings of the 20th European MPI Users' Group Meeting. EuroMPI '13. New York, NY, USA: ACM; 2013. p. 139–140.

[176] Thain D, Moretti C, Hemmes J. Chirp: a practical global filesystem for cluster and Grid computing. Journal of Grid Computing. 2009;7(1):51–72.

[177] Marozzo F, Talia D, Trunfio P. JS4Cloud: Script-based Workflow Programming for Scalable Data Analysis on Cloud Platforms. Concurrency and Computation: Practice and Experience. 2015;27(17):5214–5237.

[178] Duro FR, Marozzo F, Blas JG, et al. Exploiting in-memory storage for improving workflow executions in cloud platforms. The Journal of Supercomputing. 2016;p. 1–20.

[179] Wu X, Kumar V, Ross Quinlan J, et al. Top 10 Algorithms in Data Mining. Knowl Inf Syst. 2007 Dec;14(1):1–37.

[180] Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. Acm Sigact News. 2002;33(2):51–59.

[181] Shapiro M, Preguiça N, Baquero C, et al. A comprehensive study of convergent and commutative replicated data types. Inria–Centre Paris-Rocquencourt; INRIA; 2011.

[182] Almeida PS, Shoker A, Baquero C. Delta state replicated data types. Journal of Parallel and Distributed Computing. 2018;111:162–173.

[183] Baquero C, Almeida PS, Shoker A. Pure Operation-Based Replicated Data Types. arXiv preprint arXiv:171004469. 2017;.

[184] Martí Fraiz J. dataClay: next generation object storage. Universitat Politècnica de Catalunya; 2017. PhD dissertation.

[185] Martí J, Queralt A, Gasull D, et al. Dataclay: A distributed data store for effective inter-player data sharing. Journal of Systems and Software. 2017;131:129–145.

[186] Terry D. Replicated data consistency explained through baseball. Commun ACM. 2013;56(12):82–89. Available from: http://doi.acm.org/10.1145/2500500.

[187] Goodman JR. Cache consistency and sequential consistency. University of Wisconsin-Madison, Computer Sciences Department; 1991.

[188] Lamport L, et al. Paxos made simple. ACM Sigact News. 2001;32(4):18–25.

[189] Abadi D. Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story. IEEE Computer. 2012;45(2):37–42. Available from: https://doi.org/10.1109/MC.2012.33.

[190] Vogels W. Eventually consistent. Communications of the ACM. 2009;52(1):40–44.

[191] Lamport L. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM. 1978;21(7):558–565.

[192] DeCandia G, Hastorun D, Jampani M, et al. Dynamo: amazon's highly available key-value store. In: ACM SIGOPS operating systems review. vol. 41. ACM; 2007. p. 205–220.

[193] Davey BA, Priestley HA. Introduction to lattices and order. Cambridge university press; 2002.

[194] Vitor Enes, Carlos Baquero, Paulo Sergio Almeida, and Ali Shoker. Join Decompositions for Efficient Synchronization of CRDTs after a Network Partition. In: In the Proceedings of the ECOOP Programming Models and Languages for Distributed Computing Workshop. PMLDC'16. ACM; 2016. .

[195] Bailis P, Fekete A, Franklin MJ, et al. Coordination Avoidance in Database Systems. PVLDB. 2014;8(3):185–196. Available from: http://www.vldb.org/pvldb/vol8/p185-bailis.pdf.

[196] Careglio D, Costa GD, Ricciardi S. 2. In: Hardware Leverages for Energy Reduction in LargeâĂŘScale Distributed Systems. Wiley-Blackwell; 2015. p. 17–40. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118981122.ch2.

[197] Chen Y, Das A, Qin W, et al. Managing Server Energy and Operational Costs in Hosting Centers. SIGMETRICS Perform Eval Rev. 2005 Jun;33(1):303–314.

[198] Gschwandtner P, Knobloch M, Mohr B, et al. Modeling CPU Energy Consumption of HPC Applications on the IBM POWER7. In: Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on; 2014. p. 536–543.

[199] Hamilton J. Internet-scale Service Infrastructure Efficiency. SIGARCH Comput Archit News. 2009 Jun;37(3):232–232.

[200] Rivoire S, Ranganathan P, Kozyrakis C. A Comparison of High-level Full-system Power Models. In: Proceedings of the 2008 Conference on Power Aware Computing and Systems. HotPower'08. Berkeley, CA, USA: USENIX Association; 2008. p. 3–3.

[201] Orgerie AC, Lefevre L, Gelas JP. Demystifying energy consumption in Grids and Clouds. In: Green Computing Conference, 2010 International; 2010. p. 335–342.

[202] Wang D, Ganesh B, Tuaycharoen N, et al. DRAMsim: A Memory System Simulator. SIGARCH Comput Archit News. 2005 Nov;33(4):100–107.

[203] Kim Y, Yang W, Mutlu O. Ramulator: A Fast and Extensible DRAM Simulator. Computer Architecture Letters. 2015;PP(99):1–1.

[204] Waldspurger CA. Memory Resource Management in VMware ESX Server. SIGOPS Oper Syst Rev. 2002 Dec;36(SI):181–194. Available from: http://doi.acm.org/10.1145/844128.844146.

[205] Zhang G, Wang H, Hongwu LV, et al. A dynamic memory management model on Xen virtual machine. In: Mechatronic Sciences, Electric Engineering and Computer (MEC), Proceedings 2013 International Conference on; 2013. p. 1609–1613.

[206] Habib I. Virtualization with KVM. Linux J. 2008 Feb;2008(166). Available from: http://dl.acm.org/citation.cfm?id=1344209.1344217.

[207] Zhu Q, David FM, Devaraj CF, et al. Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In: The 10th International Conference on High-Performance Computer Architecture (HPCA-10); 2004. p. 118–129.

[208] Helmbold DP, Long DDE, Sherrod B. A Dynamic Disk Spin-down Technique for Mobile Computing. In: Proceedings of the 2Nd Annual International Conference on Mobile Computing and Networking. MobiCom '96. New York, NY, USA: ACM; 1996. p. 130–142.

[209] Colarelli D, Grunwald D. Massive Arrays of Idle Disks For Storage Archives. In: Supercomputing, ACM/IEEE 2002 Conference; 2002. p. 47–47.

[210] Greenawalt PM. Modeling power management for hard disks. In: Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1994., MASCOTS '94., Proceedings of the Second International Workshop on; 1994. p. 62–66.

[211] Alshahrani R, Peyravi H. Modeling and Simulation of Data Center Networks. In: Proceedings of the 2Nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. SIGSIM PADS '14. New York, NY, USA: ACM; 2014. p. 75–82.

[212] Hu N, Fu B, Sui X, et al. DCNSim: A Unified and Cross-layer Computer Architecture Simulation Framework for Data Center Network Research. In: Proceedings of the ACM International Conference on Computing Frontiers. CF '13. New York, NY, USA: ACM; 2013. p. 19:1–19:9.

[213] Shirayanagi H, Yamada H, Kono K. Honeyguide: A VM migration-aware network topology for saving energy consumption in data center networks. 2014 IEEE Symposium on Computers and Communications (ISCC). 2012;0:000460–000467.

[214] Zhang Y, Su AJ, Jiang G. Evaluating the impact of data center network architectures on application performance in virtualized environments. In: Quality of Service (IWQoS), 2010 18th International Workshop on; 2010. p. 1–5.

[215] De Maio V, Nae V, Prodan R. Evaluating Energy Efficiency of Gigabit Ethernet and Infiniband Software Stacks in Data Centres. In: Proceedings of the 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014). IEEE Computer Society; 2014. .

[216] Orgerie AC, Lefevre L, Guerin-Lassous I, et al. ECOFEN: An End-to-end energy Cost mOdel and simulator For Evaluating power consumption in large-

scale Networks. In: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a; 2011. p. 1–6.

[217] Pelley S, Meisner D, Wenisch TF, et al. Understanding and abstracting total data center power. Workshop on Energy-Efficient Design. 2009;.

[218] Mastelic T, Oleksiak A, Claussen H, et al. Cloud Computing: Survey on Energy Efficiency. ACM Comput Surv. 2014 Dec;47(2):33:1–33:36.

[219] Kansal A, Zhao F, Liu J, et al. Virtual Machine Power Metering and Provisioning. In: Proceedings of the 1st ACM Symposium on Cloud Computing. SoCC '10. New York, NY, USA: ACM; 2010. p. 39–50.

[220] Rong H, Zhang H, Xiao S, et al. Optimizing energy consumption for data centers. Renewable and Sustainable Energy Reviews. 2016;58:674 – 691.

[221] Ben-Itzhak Y, Cidon I, Kolodny A. Performance and Power Aware CMP Thread Allocation Modeling. In: Proceedings of the 5th International Conference on High Performance Embedded Architectures and Compilers. HiPEAC'10. Springer-Verlag; 2010. p. 232–246.

[222] Lewis AW, Tzeng NF, Ghosh S. Runtime Energy Consumption Estimation for Server Workloads Based on Chaotic Time-series Approximation. ACM Trans Archit Code Optim. 2012 Oct;9(3):15:1–15:26.

[223] Beltrame G, Palermo G, Sciuto D, et al. Plug-in of Power Models in the StepNP Exploration Platform: Analysis of Power/Performance Trade-offs. In: Proceedings of the 2004 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems. CASES '04. New York, NY, USA: ACM; 2004. p. 85–92.

[224] Itoh K, Sasaki K, Nakagome Y. Trends in low-power RAM circuit technologies. Proceedings of the IEEE. 1995;83(4):524–543.

[225] Maio VD, Kecskemeti G, Prodan R. An Improved Model for Live Migration in Data Centre Simulators. In: 2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC); 2016. p. 108–117.

[226] Sun H, Stolf P, Pierson JM. Spatio-temporal thermal-aware scheduling for homogeneous high-performance computing datacenters. Future Generation Computer Systems. 2017 fÃl'vrier;71:157–170. Available from: http://doi.org/10.1016/j.future.2017.02.005-http://oatao.univ-toulouse.fr/18918/.

[227] Capozzoli A, Primiceri G. Cooling Systems in Data Centers: State of Art and Emerging Technologies. Energy Procedia. 2015;83:484 – 493. Sustainability in Energy and Buildings: Proceedings of the 7th International Conference SEB-15.

[228] Song Z, Zhang X, Eriksson C. Data Center Energy and Cost Saving Evaluation. Energy Procedia. 2015;75:1255 – 1260. Clean, Efficient and Affordable Energy for a Sustainable Future: The 7th International Conference on Applied Energy (ICAE2015).

[229] Ham SW, Kim MH, Choi BN, et al. Simplified server model to simulate data center cooling energy consumption. Energy and Buildings. 2015;86:328 – 339.

[230] Feitelson DG. Workload modeling for computer systems performance evaluation. Cambridge University Press; 2015.

[231] Costa GD, Grange L, de Courchelle I. Modeling, classifying and generating large-scale Google-like workload. Sustainable Computing: Informatics and Systems. 2018;Available from: http://www.sciencedirect.com/science/article/pii/S2210537917301634.

[232] Feitelson DG. Resampling with FeedbackâĂŤA New Paradigm of Using Workload Data for Performance Evaluation. In: European Conference on Parallel Processing. Springer; 2016. p. 3–21.

[233] Casanova H, Legrand A, Quinson M. SimGrid: A Generic Framework for Large-Scale Distributed Experiments. In: Proceedings of the Tenth International Conference on Computer Modeling and Simulation. UKSIM '08. Washington, DC, USA: IEEE Computer Society; 2008. p. 126–131.

[234] Kliazovich D, Bouvry P, Khan S. GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing. 2012;62(3):1263–1283.

[235] Casanova H. Simgrid: A Toolkit for the Simulation of Application Scheduling. In: CCGRID. IEEE Computer Society; 2001. p. 430–441.

[236] Calheiros RN, Ranjan R, Beloglazov A, et al. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software: Practice and Experience. 2011 January;41(1):23–50.

[237] Ostermann S, Plankensteiner K, Prodan R, et al. GroudSim: An Event-Based Simulation Framework for Computational Grids and Clouds. In: Guarracino MR, Vivien F, Träff JL, et al., editors. Euro-Par 2010 Parallel Processing Workshops. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. p. 305–313.

[238] Kecskemeti G. DISSECT-CF: A simulator to foster energy-aware scheduling in infrastructure clouds. Simulation Modelling Practice and Theory. 2015;58:188 – 218. Special issue on Cloud Simulation.

[239] PiÄĚtek W, Oleksiak A, Costa GD. Energy and thermal models for simulation of workload and resource management in computing systems. Simulation Modelling Practice and Theory. 2015;58:40 – 54. Special Issue on TECHNIQUES AND APPLICATIONS FOR SUSTAINABLE ULTRASCALE COMPUTING SYSTEMS. Available from: http://www.sciencedirect.com/science/article/pii/S1569190X15000684.

[240] Meisner D, Wenisch TF. Stochastic queuing simulation for data center workloads. In: Exascale Evaluation and Research Techniques Workshop; 2010. p. 9.

[241] Liu N, Carothers C, Cope J, et al. Model and simulation of exascale communication networks. Journal of Simulation. 2012;6(4):227–236.

[242] Heinrich FC, Cornebize T, Degomme A, et al. Predicting the Energy-Consumption of MPI Applications at Scale Using Only a Single Node. In: 2017 IEEE International Conference on Cluster Computing (CLUSTER); 2017. p. 92–102.

[243] Guérout T, Monteil T, Da Costa G, et al. Energy-aware simulation with DVFS. Simulation Modelling Practice and Theory. 2013;39:76–91.

[244] Caux S, al. datazero: Deliverable D2.4 Sources and Material profiling. IRIT; 2017.

[245] Caux S, Rostirolla G, Stolf P. Smart Datacenter Electrical Load Model for Renewable Sources Management (regular paper). In: nternational Conference on Renewable Energies and Power Quality (ICREPQ), Salamanca, Spain, 21/03/18-23/03/18. vol. 16. http://www.icrepq.com: European Association for the Development of Renewable Energies, Environment and Power Quality; 2018. p. 127–132. Available from: https://doi.org/10.24084/repqj16.231.

[246] Grange L, Da Costa G, Stolf P. Green IT scheduling for data center powered with renewable energy. Future Generation Computer Systems. 2018 March;2018. Available from: https://doi.org/10.1016/j.future.2018.03.049.

[247] Solar Radiation on a Tilted Surface.. PVeducation; 2018. Available from: http://www.pveducation.org/pvcdrom/properties-of-sunlight/solar-radiation-on-a-tilted-surface.

[248] Sheme E, Holmbacka S, Lafond S, et al. Feasibility of Using Renewable Energy to Supply Data Centers in 60 Degrees North Latitude. Sustainable Computing: Informatics and Systems. 2017;p. 1âĂŞ14.

[249] Holmbacka S, Sheme E, Lafond S, et al. Geographical competitiveness for powering datacenters with renewable energy. In: Third International Workshop on Sustainable Ultrascale Computing Systems, NESUS 2016. Sofia, Bulgaria; October 2016. p. 15–22.

[250] Sheme E, Lafond S, Minarolli D, et al. Battery Size Impact in Green Coverage of Datacenters Powered by Renewable Energy: A Latitude Comparison. In: Barolli L, Xhafa F, Javaid N, et al., editors. Advances in Internet, Data & Web Technologies. Cham: Springer International Publishing; 2018. p. 548–559.

[251] Tesla Powerwall.. Wikipedia; 2017. Available from: https://en.wikipedia.org/wiki/Tesla_Powerwall.

[252] The Green500 list;. Available from: https://www.top500.org [cited 27.04.2018].

[253] Koomey J, Berard S, Sanchez M, et al. Implications of Historical Trends in the Electrical Efficiency of Computing. IEEE Annals of the History of Computing. 2011 March;33(3):46–54.

[254] Klingert S, Basmadjian R, Bunse C, et al.. Fit4green-energy aware ict optimization policies. IRIT; 2010.

[255] Bertoncini M, Pernici B, Salomie I, et al. Games: Green active management of energy in it service centres. In: Forum at the Conference on Advanced Information Systems Engineering (CAiSE). Springer; 2010. p. 238–252.

[256] Basmadjian R, Lovasz G, Beck M, et al. A generic architecture for demand response: the ALL4Green approach. In: Cloud and Green Computing (CGC), 2013 Third International Conference on. IEEE; 2013. p. 464–471.

[257] vor dem Berge M, Christmann W, Volk E, et al. CoolEmAll-Models and tools for optimization of data center energy-efficiency. In: Sustainable Internet and ICT for Sustainability (SustainIT), 2012. IEEE; 2012. p. 1–5.

[258] Wajid U, Pernici B, Francis G. Energy efficient and CO2 aware cloud computing: Requirements and case study. In: Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on. IEEE; 2013. p. 121–126.

[259] Pierson JM, al. datazero: DATAcenter with Zero Emission and RObust management using renewable energy. IRIT; 2018.

[260] Delaval G, Gueye SMK, Rutten E, et al. Modular coordination of multiple autonomic managers. In: Proceedings of the 17th international ACM Sigsoft symposium on Component-based software engineering. ACM; 2014. p. 3–12.

[261] Zhang Y, Wang Y, Wang X. GreenWare: Greening Cloud-Scale Data Centers to Maximize the Use of Renewable Energy. In: Kon F, Kermarrec AM, editors. Middleware 2011. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. p. 143–164.

[262] Andre JC, Antoniu G, Asch M, et al. Big data and extreme-scale computing: Pathways to convergence. University of Tennesse; 2018. Available from: http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/bdec2017pathways.pdf.

[263] Acton M, Bertoldi P, Booth J, et al. 2018 Best Practice Guidelines for the EU Code of Conduct on Data Centre Energy Efficiency. Joint Research Centre; 2018. Available from: http://publications.jrc.ec.europa.eu/repository/bitstream/JRC110666/kjna29103enn.pdf.

[264] Committee AT. Data Center Design and Operation. ASHRAE Datacom Series; 2014.

[265] Sartor D. Best Practices for Data Center Energy Efficiency; 2017. Available from: https://datacenters.lbl.gov/sites/default/files/Shanghai%20Data%20Center%20Dynamics%20Workshop%20060917%20%281%29.pdf.

[266] Open Data Center; 2018. Available from: http://www.opendatacenter.cn.

[267] Open Compute Project; 2018. Available from: http://www.opencompute.org/.

[268] Barrass H, Belady C, Berard S, et al. PUE: A Comprehensive Examination of the Metric. The Green Grid; 2012. Available from: https://datacenters.lbl.gov/sites/default/files/WP49-PUE%20A%20Comprehensive%20Examination%20of%20the%20Metric_v6.pdf.

[269] Barroso LA, Clidaras J, Hölzle U. The datacenter as a computer: An introduction to the design of warehouse-scale machines, second edition. vol. 24 of Synthesis Lectures on Computer Architecture; 2013.

[270] Fu RH, He ZG, Zhang X. Life cycle cost based optimization design method for an integrated cooling system with multi-operating modes. Applied Thermal Engineering. 2018;p. –. Available from: https://www.sciencedirect.com/science/article/pii/S1359431118309049.

[271] Chen H, Cheng WL, Zhang WW, et al. Energy saving evaluation of a novel energy system based on spray cooling for supercomputer center. Energy. 2017;141:304 – 315. Available from: http://www.sciencedirect.com/science/article/pii/S0360544217316080.

[272] Ndukaife TA, Nnanna AGA. Optimization of Water Consumption in Hybrid Evaporative Cooling Air Conditioning Systems for Data Center Cooling

Applications. Heat Transfer Engineering. 2018;0(0):1–15. Available from: https://doi.org/10.1080/01457632.2018.1436418.

[273] Li Z, Kandlikar SG. Current Status and Future Trends in Data-Center Cooling Technologies. Heat Transfer Engineering. 2015;36(6):523–538. Available from: https://doi.org/10.1080/01457632.2014.939032.

[274] vor dem Berge M, Costa GD, Kopecki A, et al. Modeling and Simulation of Data Center Energy-Efficiency in CoolEmAll. In: Energy Efficient Data Centers (E2DC 2012); 2012. p. 25–36.

[275] Oleksiak A, Piatek W, Kuczynski K, et al. Reducing Energy Costs in Data Centres Using Renewable Energy Sources and Energy Storage. In: Proceedings of the 5th International Workshop on Energy Efficient Data Centres. E2DC '16. New York, NY, USA: ACM; 2016. p. 5:1–5:8. Available from: http://doi.acm.org/10.1145/2940679.2940684.

[276] Center for Expertise in Energy Efficient Data Centers; 2018. Available from: https://datacenters.lbl.gov/.

[277] The Green Grid; 2018. Available from: https://www.thegreengrid.org.

[278] Asetek; 2018. Available from: https://www.asetek.com/.

[279] Aquaris^{TM} Water Cooled Cooling Solutions; 2018. Available from: https://www.aquilagroup.com/aquarius/.

[280] CoolIT; 2018. Available from: https://www.coolitsystems.com.

[281] Green Revolution Cooling; 2018. Available from: https://www.grcooling.com/.

[282] Eiland R, Fernandes JE, Vallejo M, et al. Thermal Performance and Efficiency of a Mineral Oil Immersed Server Over Varied Environmental Operating Conditions. Journal of Electronic Packaging. 2017;139:041005. Available from: https://doi.org/10.1007/s00450-016-0328-1.

[283] OpenCompute Project; 2018. Available from: http://www.opencompute.org/.

[284] Rugged POD; 2018. Available from: http://www.horizon-computing.com/?page_id=172.

[285] Tuma PE. Evaporator/boiler design for thermosyphons utilizing segregated hydrofluoroether working fluids. In: Twenty-Second Annual IEEE Semiconductor Thermal Measurement And Management Symposium; 2006. p. 69–77.

[286] Tuma PE. Fluoroketone C2F5C(O)CF(CF3)2 as a Heat Transfer Fluid for Passive and Pumped 2-Phase Applications. In: 2008 Twenty-fourth Annual IEEE Semiconductor Thermal Measurement and Management Symposium; 2008. p. 173–179.

[287] Tuma PE. Design considerations relating to non-thermal aspects of passive 2-phase immersion cooling. In: 2011 27th Annual IEEE Semiconductor Thermal Measurement and Management Symposium; 2011. p. 1–9.

[288] Campbell L, Tuma P. Numerical prediction of the junction-to-fluid thermal resistance of a 2-phase immersion-cooled IBM dual core POWER6 processor. In: 2012 28th Annual IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM); 2012. p. 36–44.

[289] Coles H, Ellsworth M, Martinez DJ. "Hot" for Warm Water Cooling. In: State of the Practice Reports. SC '11. New York, NY, USA: ACM; 2011. p. 17:1–17:10. Available from: http://doi.acm.org/10.1145/2063348.2063371.

[290] Smoyer JL, Norris PM. Brief Historical Perspective in Thermal Management and the Shift Toward Management at the Nanoscale. Heat Transfer Engineering. 2018;0(0):1–14. Available from: https://doi.org/10.1080/01457632.2018.1426265.

[291] Qouneh A, Li C, Li T. A Quantitative Analysis of Cooling Power in Container-based Data Centers. In: Proceedings of the 2011 IEEE International Symposium on Workload Characterization. IISWC '11. Washington, DC, USA: IEEE Computer Society; 2011. p. 61–71. Available from: http://dx.doi.org/10.1109/IISWC.2011.6114197.

[292] Flucker S, Tozer R. Data Centre Energy Efficiency Analysis to minimize total cost of ownership. Building Services Engineering Research and Technology. 2013;34(1):103–117. Available from: https://doi.org/10.1177/0143624412467196.

[293] Hardware Labs; 2018. Available from: http://hardwarelabs.com/.

[294] Svensson G, Södberg J. A Heat Re-Use System for the Cray XE6 and Future Systems at PDC, KTH. In: Proc. Cray User Group. Stuttgart, Germany: Cray User Group; 2012. .

[295] Romero M, Hasselqvist H, Svensson G. Supercomputers Keeping People Warm in the Winter. In: Proceedings of the 2014 conference ICT for Sustainability. Advances in Computer Science Research. Paris, France: Atlantis Press; 2014. p. 324–332. Available from: https://www.atlantis-press.com/proceedings/ict4s-14/13458.

[296] Ovaska SJ, Dragseth RE, Hanssen SA. Direct-to-chip Liquid Cooling for Reducing Power Consumption in a Subarctic Supercomputer Centre. Int J High Perform Comput Netw. 2016 Apr;9(3):242–249. Available from: http://dx.doi.org/10.1504/IJHPCN.2016.076269.

[297] Shoji F, Tanaka K, Matsushita S, et al. Improving the energy efficiencies of power supply and cooling facilities for 10 peta-scale supercomputer. Computer Science - Research and Development. 2016;31(4):235–243. Available from: https://doi.org/10.1007/s00450-016-0328-1.

[298] Coles H, Herrlin M. Immersion Cooling of Electronics in DoD Installations. Lawrence Berkeley National Laboratory; 2016. LBNL-1005666. Available from: https://datacenters.lbl.gov/sites/default/files/ImmersionCooling2016.pdf.

[299] Volk E, Rathgeb D, Oleksiak A. CoolEmAllâĂŤoptimising cooling efficiency in data centres. Computer Science - Research and Development. 2014;29(3-4):253–261. Available from: https://doi.org/10.1007/s00450-013-0246-4.

[300] Kurowski K, Oleksiak A, Piatek W, et al. DCworms - A tool for simulation of energy efficiency in distributed computing infrastructures. Simulation Modelling Practice and Theory. 2013;39:135–151. Available from: https://doi.org/10.1016/j.simpat.2013.08.007.

[301] OpenFOAM; 2018. Available from: https://openfoam.org/.

[302] OpenLB; 2018. Available from: http://optilb.org/.
[303] Palabos; 2018. Available from: https://palabos.org/.
[304] Fluent; 2018. Available from: https://www.ansys.com/Products/Fluids/ANSYS-Fluent.
[305] CoolSim; 2018. Available from: http://coolsimsoftware.com.
[306] Comsol Multiphysics; 2018. Available from: https://comsol.com/.
[307] Star-CCM+; 2018. Available from: https://mdx.plm.automation.siemens.com/star-ccm-plus.
[308] 6SigmaRoom; 2018. Available from: https://www.futurefacilities.com/products/6sigmaroom/.
[309] FlowTherm; 2018. Available from: https://www.mentor.com/products/mechanical/flotherm/.
[310] TileFlow; 2018. Available from: http://inres.com/Products/TileFlow/tileflow.html.
[311] Dakota; 2018. Available from: https://dakota.sandia.gov/.
[312] Kunkel J. Virtual Institute for I/O; 2018. Available from: https://www.vi4io.org/.
[313] Bates N. Energy Efficiency Working Group; 2018. Available from: https://eehpcwg.llnl.gov.
[314] Varrette S, Bouvry P, Cartiaux H, et al. Management of an Academic HPC Cluster: The UL Experience. In: Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014). Bologna, Italy: IEEE; 2014. p. 959–967.
[315] Emeras J, Varrette S, Bouvry P. Amazon Elastic Compute Cloud (EC2) vs. in-House HPC Platform: a Cost Analysis. In: Proc. of the 9th IEEE Intl. Conf. on Cloud Computing (CLOUD 2016). San Francisco, USA: IEEE Computer Society; 2016. .
[316] Jiang W, Liu F, Tang G, et al. Virtual Machine Power Accounting with Shapley Value. In: IEEE International Conference on Distributed Computing Systems (ICDCS); 2017. p. 1683–1693.
[317] Kurpicz M, Orgerie AC, Sobe A, et al. Energy-proportional profiling and accounting in heterogeneous virtualized environments. Sustainable Computing: Informatics and Systems. 2017;.
[318] Margery D, Guyon D, Orgerie AC, et al. A CO2 Emissions Accounting Framework with Market-based Incentives for Cloud Infrastructures. In: International Conference on Smart Cities and Green ICT Systems (SMARTGREENS); 2017. p. 299–304.
[319] Cappello F, Caron E, Dayde M, et al. Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In: The 6th IEEE/ACM International Workshop on Grid Computing, 2005.; 2005. p. 8 pp.–.
[320] O'Loughlin J, Gillam L. Towards Performance Prediction for Public Infrastructure Clouds: An EC2 Case Study. In: CloudCom 2013 IEEE. vol. 1; 2013. p. 475–480.
[321] Ostermann S, Iosup A, Yigitbasi N, et al. A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing. In: Avresky D, Diaz M,

Bode A, et al., editors. Cloud Computing. vol. 34. Springer Berlin Heidelberg; 2010. p. 115–131.

[322] Jiang C, Wang Y, Ou D, et al. Energy Proportional Servers: Where Are We in 2016? In: IEEE International Conference on Distributed Computing Systems (ICDCS); 2017. p. 1649–1660.

[323] Statista. Global electricity prices by select countries in 2017; 2017. urlhttps://www.statista.com/statistics/263492/electricity-prices-in-selected-countries/.

[324] Guyon D, Orgerie AC, Morin C. GLENDA: Green Label Towards Energy proportioNality for IaaS DAta Centers. In: International Conference on Future Energy Systems (e-Energy) Workshops; 2017. p. 302–308.

[325] Sastre J, Ibáñez JJ, Defez E, et al. Efficient scaling-squaring Taylor method for computing matrix exponential. SIAM J on Scientific Comput. 2015;37(1):A439–455.

[326] Hochbruck M, Lubich C, Selhofer H. Exponential Integrators for Large Systems of Differential Equations. The SIAM Journal on Scientific Computing. 1998 Sep;19(5):1552–1574.

[327] Higham NJ. Functions of Matrices: Theory and Computation. Philadelphia, PA, USA: SIAM; 2008.

[328] Williams DF, Hayden LA, Marks RB. A Complete Multimode Equivalent-Circuit Theory for Electrical Design. J Res Natl Inst Stand Technol. 1997;102(4):405–423.

[329] Cox SM, Matthews PC. Exponential Time Differencing for Stiff Systems. J of Comput Physics, Elsevier. 2002;176:430–455.

[330] Kassam AK, Trefethen LN. Fourth-Order Time-Stepping for Stiff PDEs. The SIAM J on Scientific Comp. 2005;26(4):1214–1233.

[331] Sastre J, Ibáñez JJ, Defez E, et al. Computing matrix functions arising in engineering models with orthogonal matrix polynomials. Mathematical and Computer Modelling. 2013;57:1738–1743.

[332] Sastre J, Ibáñez JJ, Defez E, et al. Accurate matrix exponential computation to solve coupled differential. Mathematical and Computer Modelling. 2011;54:1835–1840.

[333] Alonso P, Ibáñez J, Sastre J, et al. Efficient and accurate algorithms for computing matrix trigonometric functions. Journal of Computational and Applied Mathematics. 2017 January;309:325–332.

[334] Alonso P, Peinado J, Ibáñez J, et al. Computing matrix trigonometric functions with GPUs through Matlab. The Journal of Supercomputing. 2018 Apr;.

[335] Beaumont O, Boudet V, Legrand A, et al. Heterogeneous Matrix-Matrix Multiplication, or Partitioning a Square into Rectangles: NP-Completeness and Approximation Algorithms. In: EuroMicro Workshop on Parallel and Distributed Computing (EuroMicro'2001). IEEE Computer Society Press; 2001. p. 298–305.

[336] DeFlumere A, Lastovetsky A, Becker B, et al. Partitioning for parallel matrix-matrix multiplication with heterogeneous processors: The optimal

solution. In: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), IEEE 26th International. IEEE; 2012. p. 125–139.

[337] Tomov S, Dongarra J, Baboulin M. Towards dense linear algebra for hybrid GPU accelerated manycore systems. Parallel Computing. 2010 Jun;36(5-6):232–240.

[338] Haidar A, Dongarra J, Kabir K, et al. HPC Programming on Intel Many-Integrated-Core Hardware with MAGMA Port to Xeon Phi. Scientific Programming. 2015 January;23.

[339] Lawson CL, Hanson RJ, Kincaid DR, et al. Basic Linear Algebra Subprograms for Fortran Usage. ACM Trans Math Softw. 1979 Sep;5(3):308–323.

[340] Intel Corporation. Intel Math Kernel Library (MKL);. Accessed: 2016-04-12. Available from: http://software.intel.com/en-us/intel-mkl.

[341] OpenBLAS: An optimized BLAS library;. Accessed: 2016-04-12. http://www.openblas.net.

[342] Van Zee FG, van de Geijn RA. BLIS: A Framework for Rapidly Instantiating BLAS Functionality. ACM Transactions on Mathematical Software. 2015;41(3):14:1–14:33. Available from: https://github.com/flame/blis.

[343] NVIDIA. NVIDIA CUDA Basic Linear Algebra Subroutines (cuBLAS) library; 2016. https://developer.nvidia.com/cublas.

[344] The Open MPI Team. Open Message Passing Iterface: Open Source High Performance Computing; 2016. Accessed: 2016-04-12. https://www.open-mpi.org.

[345] Ravi Reddy Manumachu. `powerrun`: A tool to measure energy;.

[346] NVIDIA. NVML Reference Manual; 2013. https://developer.nvidia.com/nvidia-management-library-nvml.

[347] Intel Corporation. Intel Manycore Platform Software Stack (Intel MPSS); 2016. Available from: https://software.intel.com/en-us/articles/intel-manycore-platform-software-stack-mpss.

[348] Zhuo J, Chakrabarti C. Energy-efficient dynamic task scheduling algorithms for DVS systems. ACM Trans Embed Comput Syst. 2008;7(2):1–ĹŠ–ĹŠ25.

[349] Rauber T, Rünger G, Stachowski M. Model-based Optimization of the Energy Efficiency of Multi-threaded Applications. In: Proceedings of the 8th International Green and Sustainable Computing Conference (IGSC'17); 2017. .

[350] Choi J, Bedard D, Fowler RJ, et al. A Roofline Model of Energy. In: 27th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2013, Cambridge, MA, USA, May 20-24, 2013. IEEE Computer Society; 2013. p. 661–672. Available from: https://doi.org/10.1109/IPDPS.2013.77.

[351] Rauber T, Rünger G. How do loop transformations affect the energy consumption of multi-threaded Runge-Kutta methods? In: Proceedings of the 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing; 2018. .

[352] Cabrera A, Almeida F, Blanco V, et al. Analytical Modeling of the Energy Consumption for the High Performance Linpack. In: 2013 21st Euromi-

cro International Conference on Parallel, Distributed, and Network-Based Processing; 2013. p. 343–350.

[353] Cabrera A, Almeida F, Arteaga J, et al. Measuring energy consumption using EML (energy measurement library). Computer Science - Research and Development. 2015 May;30(2):135–143. Available from: https://doi.org/10.1007/s00450-014-0269-5.

[354] Zaslavsky GM. Chaos, fractional kinetics, and anomalous transport. Physics Reports. 2002;371(6):461–580.

[355] EringenBates AC. Nonlocal Continuum Field Theories. Springer; 2002.

[356] Silling SA. Reformulation of elasticity theory for discontinuities and long-range forces. Journal of the Mechanics and Physics of Solids. 2000;48(1):175–209.

[357] Bakunin OG. Turbulence and Diffusion: Scaling Versus Equations. Springer; 2008.

[358] McCay BM, Narasimhan MNL. Theory of nonlocal electromagnetic fluids. Archives of Mechanics. 1981;33(3):365–384.

[359] Ainsworth M, Glusa C. Aspects of an adaptive finite element method for the fractional Laplacian: A priori and a posteriori error estimates, efficient implementation and multigrid solver. Comput Methods Appl Mech Engrg. 2017;327:4–35.

[360] Bear J, Cheng AHD. Modeling Groundwater Flow and Contaminant Transport. Springer; 2010.

[361] Bates PW. On some nonlocal evolution equations arising in materials science. Nonlinear dynamics and evolution equations. 2006;48:13–52.

[362] Zijlstra ES, Kalitsov A, Zier T, et al. Fractional Diffusion in Silicon. Advanced Materials. 2013;25(39):5605–5608.

[363] Bueno-Orovio A, Kay D, Grau V, et al. Fractional diffusion models of cardiac electrical propagation: role of structural heterogeneity in dispersion of repolarization. Journal of the Royal Society Interface. 2014;11(97).

[364] Harizanov S, Margenov S, Marinov P, et al. Volume constrained 2-phase segmentation method for utilizing a linear system solver based on the best uniform polynomial approximation of $x^{-1/2}$. Journal of Computational and Applied Mathematics. 2017;310:115–128.

[365] Gilboa G, Osher S. Nonlocal operators with applications to image processing. Multiscale Modeling & Simulation. 2008;7(3):1005–1028.

[366] Abirami A, Prakash P, Thangavel K. Fractional diffusion equation-based image denoising model using CNŰGL scheme. Int J Computer Mathematics. 2017;p. 1–18.

[367] Kilbas AA, Srivastava HM, Trujillo JJ. Theory and Applications of Fractional Differential Equations. Elsevier; 2006.

[368] Pozrikidis C. The Fractional Laplacian. Chapman and Hall/CRC; 2016.

[369] Čiegis R, Starikovičius V, Margenov S. On parallel numerical algorithms for fractional diffusion problems. In: Proceedings of the Third International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2016).

IICT-BAS, Sofia, Bulgaria; October 6–7, 2016. p. 85–90. NESUS, ICT COST Action IC1305.

[370] Čiegis R, Starikovičius V, Margenov S, et al. Parallel solvers for fractional power diffusion problems. Concurrecy Comput: Pract Exper. 2017;25(24).

[371] Čiegis R, Starikovičius V, Margenov S, et al. A comparison of accuracy and efficiency of parallel solvers for fractional power diffusion problems. In: Parallel Processing and Applied Mathematics, (PPAM2017, Lublin, Poland, September 9–13, 2017) Proceedings, part I. vol. 10777 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2018. p. 000–000.

[372] Nochetto RH, Otárola E, Salgado AJ. A PDE approach to fractional diffusion in general domains: a priori error analysis. Foundations of Computational Mathematics. 2015;15(3):733–791.

[373] Vabishchevich P. Numerical solving unsteady space-fractional problems with the square root of an elliptic operator. Mathematical Modelling and Analysis. 2016;21(2):220–238.

[374] Bonito A, Pasciak JE. Numerical approximation of fractional powers of elliptic operators. Mathematics of Computation. 2015;84(295):2083–2110.

[375] Harizanov S, Lazarov R, Marinov P, et al. Optimal solvers for linear systems with fractional powers of sparse SPD matrices. Numerical Linear Algebra with Applications. 2018;.

[376] Napov A, Notay Y. An Algebraic Multigrid Method with Guaranteed Convergence Rate. SIAM Journal on Scientific Computing. 2012;34(2):A1079–A1109.

[377] Falgout R, Yang U. Hypre: A library of high performance preconditioners. In: Computational Science 2002. International Conference (ICCS, Amsterdam, The Netherlands, April 21–24, 2002) Proceedings, part III. vol. 2331 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2002. p. 632–641.

[378] Falgout R, Jones J, Yang U. The design and implementation of Hypre, a library of parallel high performance preconditioners. In: Numerical Solution of Partial Differential Equations on Parallel Computers, part III. vol. 51 of Lecture Notes in Computational Science and Engineering. Springer, Berlin, Heidelberg; 2006. p. 264–294.

[379] Bonito A, Lei W, Pasciak JE. The approximation of parabolic equations involving fractional powers of elliptic operators. Journal of Computational and Applied Mathematics. 2017;315:32–48.

[380] Bonito A, Borthagaray J, Nochetto RH, et al. Numerical methods for fractional diffusion. Computing and Visualization in Science. 2017;000:00–00.

[381] Lazarov R, Vabishchevich P. A numerical study of the homogeneous elliptic equation with fractional order boundary conditions. Fractional Calculus and Applied Analysis. 2017;20(2):337–351.

[382] Acosta G, Borthagaray J. A Fractional Laplace Equation: Regularity of Solutions and Finite Element Approximations. SIAM Journal on Numerical Analysis. 2017;55(2):472–495.

[383] Vabishchevich PN. Numerically Solving an Equation for Fractional Powers of Elliptic Operators. Journal of Computational Physics. 2015;282:189–302.

[384] Čiegis R, Starikovičius V, Tumanova N, et al. Application of distributed parallel computing for dynamic visual cryptography. The Journal of Supercomputing. 2016;72(11):4204–4220.

[385] Nochetto R, Otárola E, Salgado A. A PDE approach to numerical fractional diffusion. In: Proceedings of the 8th ICIAM, Beijing, China; 2015. p. 211–236.

[386] Quarteroni A, Valli A. Domain Decomposition Methods for Partial Differential Equations. Oxford Science Publications; 1999.

[387] Čiegis R, Tumanova N. On construction and analysis of finite difference schemes for pseudoparabolic problems with nonlocal boundary conditions. Mathematical Modelling and Analysis. 2014;19(2):281–297.

[388] Varga RS, Carpenter AJ. Some numerical results on best uniform rational approxiumation of x^α on $[0, 1]$. Numerical Algorithms. 1992;2(2):171 – 185.

[389] Amiranashvili S, Čiegis R, Radziunas M. Numerical methods for a class of generalized nonlinear Schrödinger equations. Kinetic and Related Models. 2015;8(2):215–234.

[390] Hadamard J. Sur les Problemes aux Derivees Partielles et leur Signification Physique. Bull Princeton University; 1902.

[391] Lowrie W. Fundamentals of Geophysics. Cambridge University Press; 2007.

[392] Sen M, Stoffa P. Global Optimization Methods in Geophysical Inversion. Elsevier Science B.V.; 1995.

[393] Xiaobing Z. Analytic solution of the gravity anomaly of irregular 2D masses with density contrast varying as a 2D polynomial function. Geophysics. 2010;75(2):I11–I19.

[394] Xiaobing Z. 3D vector gravity potential and line integrals for the gravity anomaly of a rectangular prism with 3D variable density contrast. Geophysics. 2009;74(6):I43–I53.

[395] Silva J, Medeiros WE, Barbosa VCF. Gravity inversion using convexity constraint. Geophysics. 2000;65(1):102–112.

[396] 3D stochastic inversion of borehole and surface gravity data using geostatistics. EGM International Workshop on Adding new Value to Electromagnetic, Gravity and Magnetic Methods for Exploration, Capri, Italy, April 11-14; 2010.

[397] Wellmann FJ, Horowitz FG, Schill E, et al. Towards incorporating uncertainty of structural data in 3D geological inversion. Elsevier Tectonophysics TECTO-124902; 2010.

[398] Zhdanov MS, Wilson GA, Xiaojun L. 3D imaging of subsurface structures using migration and regularized focusing inversion of gravity and gravity gradiometry data. Airborne Gravity - Abstracts from the ASEG-PESA Airborne Gravity Workshop, Geoscience Australia Record. 2010;23.

[399] Kiflu H, Kruse S, Loke MH, et al. Improving resistivity survey resolution at sites with limited spatial extent using buried electrode arrays. Journal of Applied Geophysics. 2016;135(Oct).

[400] Rickwood P, Sambridge M. Efficient parallel inversion using the neighborhood algorithm. Geochemistry Geophysics Geosystems - Electronic Journal of the Earth Sciences. 2006;7(11).

[401] Loke MH, Wilkinson P. Rapid parallel computation of optimized arrays for electrical imaging surveys. Near Surface 2009 - 15th European Meeting of Environmental and Engineering Geophysics, Dublin, Ireland, 7-9 Sept. 2009;p. B11.

[402] Zuzhi H, Zhanxiang H, Yongtao W, et al. Constrained inversion of magnetotelluric data using parallel simulated annealing algorithm and its application. In: EM P4 Modeling and Inversion, v.29. SEG Denver Annual Meeting - SEG Expanded Abstracts; 2010. p. 895–899.

[403] Wilson G, Ĉuma M, Zhdanov MS. Massively parallel 3D inversion of gravity and gravity gradiometry data. PREVIEW, The Magazine of the Australian Society of Exploration Geophysicists. 2011;June.

[404] Högbom JA. Aperture synthesis with a non-regular distribution of interferometer baselines. Astr Astrophys Suppl. 1974;15:417.

[405] Frasheri N, Bushati S. An algorithm for gravity anomaly inversion in HPC. SCPE: Scalable Computing: Practice and Experience. 2012;13(2):51–69.

[406] Frasheri N, Cico B. Analysis of the convergence of iterative gravity inversion in parallel systems. In: Kocarev L, editor. Springer Advances in Intelligent and Soft Computing 150: ICT Innovations 2011. Springer-Verlag; 2012. p. 219–222.

[407] Frasheri N, Cico B. Scalability of geophysical inversion with OpenMP and MPI in parallel processing. In: Markovski S, Gusev M, editors. Springer Advances in Intelligent Systems and Computing 207: ICT Innovations 2012: Secure and Intelligent Systems. Springer-Verlag; 2013. p. 345–352.

[408] A parallel processing algorithm for gravity inversion. European Geosciences Union General Assembly EGU'2013 Vienna 7-12 April; 2013. Available from: https://meetingorganizer.copernicus.org/EGU2013/EGU2013-8739-1.pdf.

[409] Brélaz D. New methods to color the vertices of a graph. Communications of the ACM. 1979;22(4):251–256.

[410] Culberson J. Iterated greedy graph coloring and the difficulty landscape. University of Alberta; 1992.

[411] Sloane N. Challenge problems: Independent sets in graphs. Information Sciences Research Center, http://neilsloanecom/doc/graphshtml. 2005;.

[412] Hasselberg J, Pardalos PM, Vairaktarakis G. Test case generators and computational results for the maximum clique problem. Journal of Global Optimization. 1993;3(4):463–482.

[413] Szabó S. Monotonic matrices and clique search in graphs. Annales Univ Sci Budapest, Sect Comp. 2013;41:307–322.

[414] Szabó S, Zaválnij B. Benchmark Problems for Exhaustive Exact Maximum Clique Search Algorithms. In: Middle-European Conference on Applied Theoretical Computer Science (MATCOS 2016): Proceedings of the 19th International Multiconference. Information Society – IS; 2016. p. 65–67.

[415] Karger D, Motwani R, Sudan M. Approximate graph coloring by semidefinite programming. Journal of the ACM (JACM). 1998;45(2):246–265.

[416] Godsil C, Royle GF. Algebraic graph theory. vol. 207. Springer Science & Business Media; 2013.

[417] Elphick C, Wocjan P. An inertial lower bound for the chromatic number of a graph. arXiv preprint arXiv:160501978. 2016;.

[418] Östergård PR. A fast algorithm for the maximum clique problem. Discrete Applied Mathematics. 2002;120(1-3):197–207.

[419] Li CM, Quan Z. An Efficient Branch-and-Bound Algorithm Based on MaxSAT for the Maximum Clique Problem. In: Proc. of AAAI. vol. 10; 2010. p. 128–133.

[420] Li CM, Fang Z, Xu K. Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem. In: Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on. IEEE; 2013. p. 939–946.

[421] San Segundo P, Nikolaev A, Batsyn M. Infra-chromatic bound for exact maximum clique search. Computers & Operations Research. 2015;64:293–303.

[422] Szabó S, Zaválnij B. Reducing graph coloring to clique search. Asia Pacific Journal of Mathematics. 2016;3(1):64–85.

[423] Li CM, Quan Z. Combining graph structure exploitation and propositional reasoning for the maximum clique problem. In: Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on. vol. 1. IEEE; 2010. p. 344–351.

[424] Fu Z, Malik S. On solving the partial MAX-SAT problem. In: International Conference on Theory and Applications of Satisfiability Testing – SAT2006. Lecture Notes in Computer Science; 2006. p. 252–265.

[425] Lovász L. On the Shannon capacity of a graph. IEEE Transactions on Information theory. 1979;25(1):1–7.

[426] Borchers B. CSDP, A C library for semidefinite programming. Optimization methods and Software. 1999;11(1-4):613–623.

[427] Borchers B, Young JG. Implementation of a primal–dual method for SDP on a shared memory parallel architecture. Computational Optimization and Applications. 2007;37(3):355–369.

[428] Carraghan R, Pardalos PM. An exact algorithm for the maximum clique problem. Operations Research Letters. 1990;9(6):375–382.

[429] Balas E, Yu CS. Finding a maximum clique in an arbitrary graph. SIAM Journal on Computing. 1986;15(4):1054–1068.

[430] Zavalnij B. Speeding up Parallel Combinatorial Optimization Algorithms with Las Vegas Method. In: 10th International Conference on Large-Scale Scientific Computing. Lecture Notes in Computer Science; 2015. p. 258–266.