energies

MDPI

Review

Energy-Aware Scheduling for High-Performance Computing Systems: A Survey

Bartłomiej Kocot [1], Paweł Czarnul [1] and Jerzy Proficz [2,*]

[1] Department of Computer Architecture, Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, 80-233 Gdańsk, Poland; bartlomiejkocot98@gmail.com (B.K.); pczarnul@eti.pg.edu.pl (P.C.)
[2] Centre of Informatics—Tricity Academic Supercomputer & Network (CI TASK), Gdansk University of Technology, 80-233 Gdańsk, Poland
* Correspondence: j.proficz@task.gda.pl

Abstract: High-performance computing (HPC), according to its name, is traditionally oriented toward performance, especially the execution time and scalability of the computations. However, due to the high cost and environmental issues, energy consumption has already become a very important factor that needs to be considered. The paper presents a survey of energy-aware scheduling methods used in a modern HPC environment, starting with the problem definition, tackling various goals set up for this challenge, including a bi-objective approach, power and energy constraints, and a pure energy solution, as well as metrics related to the subject. Then, considered types of HPC systems and related energy-saving mechanisms are described, from multicore-processors/graphical processing units (GPU) to more complex solutions, such as compute clusters supporting dynamic voltage and frequency scaling (DVFS), power capping, and other functionalities. The main section presents a collection of carefully selected algorithms, classified by the programming method, e.g., machine learning or fuzzy logic. Moreover, other surveys published on this subject are summarized and commented on, and finally, an overview of the current state-of-the-art with open problems and further research areas is presented.

Keywords: high-performance computing; energy-aware scheduling; energy-aware metrics; DVFS; power capping

Citation: Kocot, B.; Czarnul, P.; Proficz, J. Energy-Aware Scheduling for High-Performance Computing Systems: A Survey. *Energies* **2023**, *16*, 890. https://doi.org/10.3390/en16020890

Academic Editor: Edmund Widl

Received: 15 December 2022
Revised: 5 January 2023
Accepted: 9 January 2023
Published: 12 January 2023

1. Introduction

Traditionally, high-performance computing [1] has been primarily performance oriented, firstly increasing in performance and size by adding more (cluster) nodes and increasing central processing unit (CPU) frequency, then increasing the number of cores when the achievable clock frequency reached its plateau. Finally, when accelerators such as GPUs specifically well suited for running massively parallel ideally sequential instruction sequences by thousands of threads were introduced, they also allowed increasing performance/watt. Currently building new HPC systems very much considers power requirements [2] with power per system such as: 21.1 MW for the 1.102 EFlop/s Frontier, 29.9 MW for the 442 PFlop/s Fugaku, and only 2.94 MW for the 151.9 PFlop/s LUMI systems at the top of the current TOP500 list (https://www.top500.org/lists/top500/—TOP500: a list of the 500 most powerful non-distributed computer systems in the world, accessed: 15 December 2022). Moreover, the current increases in electricity costs have led researchers to optimize computations with consideration of both power (such as maximum power required through computations important when the maximum power load of a (sub)network is crucial) or total energy used or consideration of mixed performance and power/energy metrics [3].

Consequently, in view of these developments and conditions, in this paper we aim at investigating energy-aware scheduling and optimization in high-performance computing by considering and analyzing several aspects in this area, i.e.,:

- energy-aware metrics [4–6];
- types of systems in the context of their heterogeneity as well as compute device types including multi- and many-core CPUs and accelerators such as GPUs [1];
- algorithmic approaches used to solve the energy-aware related scheduling problems [7,8];
- additional aspects such as filtering data and accuracy of measurements [9,10].

The methodology adopted for the selection of research works for analysis included consideration of papers from the recent 15 years available on the Google Scholar platform (which contains a wider selection of research type of works than the highly regarded Web of Science and Scopus databases) that consider and apply energy-aware scheduling in the field of high-performance computing in a context wider than tailored to one specific application. Specifically, this applies to those that propose an approach applicable to scenarios and applications following a universal model, framework applicable to a class of codes, etc. We analyzed up to 100 papers returned for combinations of keywords including energy/energy-aware/power + scheduling + high performance computing/HPC/parallel computing. We subsequently considered metrics that are part of the optimization goal, compute device types and environments, and specific algorithms used to solve the stated goal. This allowed us to perform an investigation of algorithm $\times$ optimization metric $\times$ system type, resulting in the identification of research gaps and open problems in energy-aware HPC scheduling. We shall note that we exclude works focusing on cloud-specific research problems and technologies but at the same time, we realize that HPC solutions can and are deployed within clouds and HPC-level energy-aware computing are within our interests.

The outline of the paper is as follows. In Section 2, we review existing surveys concerning the topic. We discuss the definition of energy-aware scheduling for HPC systems and related concepts such as resource allocation, data partitioning as well as (workflow) scheduling in Section 3. We then discuss metrics and criteria considered for optimization as well as mechanisms allowing obtaining various performance-power/energy configurations such as DVFS and power capping. This leads us to review existing energy-aware scheduling approaches by the following device/system types: CPU, GPU, hybrid CPU+GPU, and finally, homo-/heterogeneous clusters. In Section 4, we proceed with analyzing algorithms used to solve the energy-aware scheduling problems by type and formulation. In Section 5, we follow with conclusions, open problems, and areas for further research that stem from our analysis.

2. Related Review Works

Work [3] provides a broad overview of articles on energy-aware tools, techniques, and environments used in high-performance computing. The work describes energy management tools for various architectures. It lists approaches for various types of architectures—a single device, cluster, grid, and cloud. The work describes the optimization metrics used. It also aligns metrics with energy-aware tools and presents benchmarking, prediction, and simulation tools for the described problem. The authors indicate the need for the unification of energy management interfaces on different architectures. Attention is drawn to the need for the development of performance and energy models for CPU-GPU systems instead of empirical approaches. The conclusions show that further development of energy-aware methods for heterogeneous environments is needed. The authors indicated optimization goals worth investigating, based on minimizing the product of energy and time. It was also suggested that there is a need for building tools for automatic power-performance tradeoffs and auto-configurable power capping. The need for validation of the quality of energy tools was also mentioned.

Article [11] analyzes currently explored areas in the field of energy-aware HPC and clouds. It categorizes the methods used to save energy for the HPC systems and analyzes

the taxonomy of energy-aware HPC computing. The article also concludes the open areas of research in the reviewed topic. Finally, several open questions on the development of energy-aware HPC computing are presented. The authors pose a question about the method of monitoring distributed systems in the context of energy (which can be carried out from cores to entire clusters). They also claim that there are not enough energy models estimating power consumption for programs, which translates into the inability to accurately estimate the cost for the user. They point out the need for the development of energy-aware tools for different levels of the HPC system (to prevent possible deviations at different levels). It also shows the lack of transparent business interfaces that allow for easier management of tradeoffs between energy and performance.

Article [12] describes possible methods of analyzing the performance of HPC systems, taking into account energy consumption. Additionally, it lists and describes the tools that are available for HPC systems for energy-efficiency analysis. It defines the most important characteristics required for energy measurement tools. The work also carried out several experiments with energy monitoring tools. The tools were classified in the context of the overhead of the tool, portability, accuracy, code improvement suggestions, and interface friendliness.

Review [7] examines resource allocation and energy-efficient algorithms for job scheduling in parallel systems. In the work, the listed solutions were simulated and compared using selected metrics (queue and wait time, slow down, and energy consumption). Simulations were carried out in the work, comparing 13 algorithms and showing their pros and cons. An analysis of workloads was performed, categorizing them into Narrow, Wide, Short, and Long jobs. It also analyzes selecting the appropriate heuristic for a given problem. The paper concludes that energy is the dominant cost in maintaining HPC systems. The authors, after analyzing many scheduling and resource allocation algorithms, conclude that to obtain the best energy savings, planning policies should be used dynamically depending on the situation.

Survey [8] describes the problems and challenges of energy-aware scheduling and resource allocation. Many types of architecture are taken into account, such as single-core systems, multicore systems, and distributed systems. The work reviews resource allocation algorithms, scheduling algorithms, and energy-saving tools. Additionally, it contributed to the field of taxonomy. The conclusions allowed them to indicate the lack of solutions in the following areas. It is shown that most algorithms assume that all resources are available during the entire planning phase, which is not always true. The authors also point out that tradeoffs other than execution time and energy have been only slightly explored. Attention is drawn to the relationship between temperature and energy. Another conclusion is that algorithms based on slack allocation may perform better for workflow tasks.

In [13] Maiterth et al. present energy-savings efforts focused on energy- and power-aware job scheduling and resource allocation, performed in nine large HPC centers located over three continents. The survey is based on a questionnaire consisting of eight questions answered by the data centers' staff. The described solutions are practical procedures introduced in the centers, including power capping, job killing, splitting compute nodes by virtual machines, etc. The paper is valuable from the engineering point of view, but it does not provide a more formal and theoretical review of the power-aware scheduling problem.

An interesting perspective of green data centers functioning is presented in survey [14]. In comparison to our work, it focuses on thermal and cooling aspects of the task scheduling, where the proper distribution of the heat in server racks is the objective. The paper presents the thermal modeling as well as possible solutions to keep the server room intact, without some hard to cool or even dangerous phenomena such as hot spots. The proposed methods are based on two approaches: reactive, where the spotted problem is resolved *a posteriori*, and proactive, where the problem is going to be avoided before its occurrence, e.g., using a thermal model of the server room followed by the proper task assignment to a compute node.

3. Problem Formulation

This section presents the problem definitions of energy-aware scheduling for HPC systems and related problems—resource allocation, data partitioning, and workflow scheduling. The symbols used in these and further formulas are listed in Table 1.

Table 1. Symbols used in the formulas.

Symbol	Description
J	set of tasks/jobs
R	set of resources
j	task/job
r	resource
$time(j,r)$	execution time of task j on resource r
$energy(j,r)$	energy needed by task j on resource r
s_j	start time of task j
a_j	resource assigned to task j
$communication(x,y,a_x,a_y)$	communication time between resources a_x, a_y performing tasks x and y
D	set of the precedence pairs
A_j	set of the resources assigned to task j
$ExecT$	execution time
EC	energy consumption
α, β	factors
W	weights
PpW	performance per watt
$RPpW$	reference performance per watt
TGI	The Green Index
REE	relative energy efficiency
$EDP(EC, ExecT)$	energy delay product
$EDS(EC, ExecT)$	energy delay summation
$EDD(EC, ExecT)$	energy delay distance

Let J be a finite set of tasks (sometimes also called jobs) and R be a finite set of resources to be used. Let $time(j,r)$ be a function that returns the execution time of job $j \in J$ on resource $r \in R$, and $energy(j,r)$ will be the function that returns the energy needed by job $j \in J$ on $r \in R$ [15,16]. Energy-aware scheduling consists of finding the sequence of start times of the tasks $\{s_1, s_2, ..., s_{|J|}\}$ and assigned resources to the tasks $\{a_1, a_2, ..., a_{|J|}\}$ in such a way that [17]:

$$\forall_{s_x} : \neg\exists_{s_y} : s_x \leq s_y + time(y, a_y) \wedge s_y \leq s_x + time(x, a_x) \wedge a_x = a_y$$

$$\forall_{a_x} : a_x \subset R$$

optimizing:

$$min/max(OptimizationGoal(\{s_1, s_2, ...s_{|J|}\}, \{a_1, a_2, ...a_{|J|}\}))$$

The optimization goal is minimized/maximized depending on the formulation of a function that involves basic metrics such as execution time, energy, and power. Additionally, constraints involving such metrics might be added as necessary conditions.

In other words, it is a system for allocating time slots of available resources to corresponding tasks and maximizing or minimizing (depending on the objective) the selected energy-efficiency goal. Both the takeoff time and the number of resources are limited. Only one job can be processed on each resource at a time in this formulation [18,19]. Depending on the system, the resources can be specific cores, CPU devices, dedicated accelerators, or entire compute nodes. The workload can also be defined in the form of a workflow defined as a directed acyclic graph (DAG) [20,21], where nodes represent the tasks and edges represent precedence relation, which imposes further constraints on the scheduler [22]. Such dependencies can be included in the set of pairs indicating that the start time of a

given task is greater than the completion time of the second task (hereafter referred to as the `D` set). The scheduling formula would be updated with the following constraint [17]:

Let us define a function $communication(x, y, a_x, a_y)$ that returns the communication time between the devices a_x and a_y performing tasks x and y. Let D be a finite set of pairs of tasks, where:

$$\forall\{x, y\} \in D : s_x + time(x, a_x) + communication(x, y, a_x, a_y) \leq s_y$$

optimizing:

$$min/max(OptimizationGoal(\{s_1, s_2, ...s_{|J|}\}, \{a_1, a_2, ...a_{|J|}\}, D)$$

The above condition can also be used to prioritize tasks for scheduling issue [23].

Occasionally, the resources that can be allocated to a job may be limited to a subset of the set of all devices [24]. Furthermore, the number of resources allocated to a task can be greater than one [25]. In this case, we assume that the `time` and `communication` functions take a set of assignments instead of values as parameters. Let P_j be a set of devices that can be assigned to task $j \in J$. The goal is to find the assignment set A_j (instead of a sequence $\{a_1, a_2, ..., a_{|J|}\}$) and start times $\{s_1, s_2, ..., s_{|J|}\}$ for each job in such a way that:

$$\forall x \in A_j : x \in P_j$$

$$\forall_{s_x} : \neg\exists_{s_y} : s_x \leq s_y + time(y, A_y) \wedge s_y \leq s_x + time(x, A_x) \wedge A_x \cap A_y \neq \emptyset$$

$$\forall\{x, y\} \in D : s_x + time(x, A_x) + communication(x, y, A_x, A_y) \leq s_y$$

optimizing:

$$min/max(OptimizationGoal(\{s_1, s_2, ..., s_{|J|}\}, A_1, ..., A_{|J|}, D)$$

In other words, it is the process of selecting available resources for a task. Resources must fulfill job processing requirements (both the software and hardware) and meet performance expectations. In this case, we also take care to process the task with the lowest possible power level, which will not disturb the imposed restrictions. Such assigned resources to tasks are often defined as resource allocation [26]. The approach is widely used in cloud computing environments [27].

The data partitioning problem overlaps with scheduling and resource allocation problems. Data partitioning is an isomorphic problem of resource allocation, but instead of allocating resources to tasks, we allocate data to resources. In this problem, we assume that we are solving a problem that is partitioned among many resources [28].

Energy-aware scheduling, resource allocation, and data partitioning can also be formulated using well-known techniques such as linear integer programming and genetic algorithm (GA) [29,30]. For this purpose, it is necessary to define suitable optimization goals.

3.1. Optimization Goals

Energy-aware optimization can be either focused on energy consumption (EC) only or can take under consideration other factors, usually the performance of the underlying systems (e.g., execution time (ExecT) of scheduled tasks). The former can be simply stated as energy itself (e.g., in joules or kWh) or can be expressed in a more sophisticated form such as in [31], with a metric described as instructions per joule equivalent or performance per watt (PpW), where it was used for efficiency setting out the Koomey's law, showing Moore's law time progression of the manufactured hardware. This formulation, in a FLOPS/watt form, is also used as a primary objective of the supercomputer evaluation in Green500 list (https://www.top500.org/lists/green500/—GREEN500: a list of the 500 most powerful non-distributed computer systems in the world, energy-aware view, accessed: 15 December 2022). Another variation of the metric is its relative version proposed in [4]: The Green

Index (TGI). It uses PpW by dividing the evaluated configuration by the reference solution to obtain the relative energy efficiency, also called PoweerUp [32]:

$$REE = \frac{PpW}{RPpW}$$

Such a metric is measured for different benchmarks and then, to obtain TGI, it is summed up with the specific weights, defined according to the importance of the benchmarks:

$$TGI = \sum (W_i \cdot REE_i)$$

In the case of considering the performance factors, we need to cope with multi-objective optimization. A typical approach used in multi-objective optimization is to combine the objectives into one meta-objective by providing a specific goal (i.e., the fitness function), reflecting their importance to the user. In this case, one of the most popular functions is the energy delay product (EDP), which is defined as a result of the multiplication of time and energy used to complete the computational task [33]. The variations of this function cover different degrees of time power:

$$EDP(EC, ExecT) = EC * ExecT^w$$

where w have values of 1, 2 or 3 [5,6].

In [34], the authors claim that the EDP does not allow for a precise definition of the tradeoff between the energy and performance—showing a list of their criteria for its evaluation, including such characteristics as stability or intuitiveness. Thus, they propose other energy-aware functions: energy delay summation (EDS) and energy delay distance (EDD). The former can be defined as the sum of energy cost and execution time multiplied by the factors that limit the impact of its components:

$$EDS(EC, ExecT) = \alpha * EC + \beta * ExecT$$

The latter (EDD) is formulated as a Euclidean distance from the most optimal point to the currently measured one:

$$EDD(EC, ExecT) = \sqrt{EC^2 + (\beta * ExecT)^2}$$

3.2. Mechanisms Allowing Performance Energy Configurations

Energy-aware scheduling algorithms usually exploit different hardware/software configurations, where the same job demonstrates different power-performance tradeoffs, e.g., heterogeneity of the used environment, with more and less energy efficient compute units. To generate even more possible configurations, it is possible to use some power-limiting mechanisms. Increasing the number of configurations leads to a larger space for searching for an appropriate plan for the scheduler. By attempting to maximize or minimize optimization goals, power tradeoffs can be made at the expense of performance. The dynamic power management (DPM), dynamic voltage and frequency scaling (DVFS), and power capping mechanisms, which can be applied dynamically or statically, are presented.

3.2.1. DPM

Dynamic power management (DPM) allows temporarily lowering the frequency or turning off a node during idle states [35]. DPM can be used for many elements of the HPC system (e.g., disks and processors). Restoring resources to service is cost-effective in performance [36]. For this reason, the scheduler is the right place to apply DPM because of its predictability. This makes it possible to extend the scheduler with further energy mechanisms.

3.2.2. DVFS

Dynamic voltage and frequency scaling (DVFS) is a method of power control by regulating the voltage and frequency of the device. This mechanism has been adapted to the energy-aware schedulers for additional energy management [37]. DVFS can be applied with tools such as CPUfreq, the interface of which allows for limiting and dynamically setting the frequency of the CPU or disabling a particular core. Since DVFS itself is not accurate enough, additional mechanisms were introduced that together result in power capping [38]. To achieve higher accuracy, power models are used to calculate appropriate parameters for the DVFS dynamically [39,40].

3.2.3. Power Capping

Power capping can be used for a particular device by setting a power threshold, which may not be exceeded by the device [41]. Device manufacturers provide appropriate tools to apply power capping, such as NVIDIA System Management Interface (nvidia-smi), RAPL (Intel), Energyscale (IBM), and APM (AMD) [42]. They also allow limiting not only the main computing units but also such components as DRAM (RAPL) [43]. Tools such as RAPL use the DVFS mechanism and clock throttling. The tool additionally monitors power consumption to dynamically adjust frequency scaling. RAPL also uses energy consumption prediction models to achieve the highest possible accuracy [38]. The interface of tools such as RAPL provides several functionalities, such as long-term/short-term limits averaging windows (dynamically adjusted), information about the maximum and minimum power consumption that can be set on the device, or control of the current energy consumption. Various power capping algorithms are under intense investigation [44]. The use of power capping and its effects on performance are also being researched [45].

3.3. Energy-Aware Scheduling for Particular System Types

Scheduling can be defined at many levels, starting from the management of particular threads, through controlling individual computing devices, such as nodes gathered within a cluster, either based directly on the hardware or virtualized in the cloud, to computational grids—federated clusters, usually located in different geographical locations. Moreover, in the case of heterogeneous systems, devices may differ in their architectures and the way of their programming such as APIs. Some of the jobs, as mentioned in Section 3, can only be performed on a particular architecture. In the case of energy-aware scheduling, the different energy consumption of the various devices has to be taken into account. The approaches of scheduling used for various combinations of systems are discussed below.

3.3.1. CPU

The use of multiple CPUs, each of which can handle multiple threads for multiple tasks simultaneously, requires appropriate planning [46]. Technologies such as power capping or DVFS allow the frequency of the cores to be changed dynamically, allowing additional configurations for energy-aware scheduling [11,23,47]. With tools such as RAPL (Intel) or APM (AMD) it is possible to create more power configurations with power capping [48–50].

The RAPL driver is capable of controlling and monitoring power domains including PP0—CPU cores, DRAM—memory, and PKG—representing the whole CPU socket [10]. DPM can be also used to reduce power consumption during idle state [35]. Processor cores can be heterogeneous and include energy-efficient and higher performance units. It can also be a subject of scheduling [51]. Another way to extend available configurations and manage possible resources is virtualization, which is especially used in the cloud (not addressed in this article) [52–55].

3.3.2. GPU

Currently, the considerable resources of GPU cards available in HPC centers that many users work with require queuing systems or schedulers to allow efficient operation of the devices. For this reason, various methods are being investigated to use these types

of accelerators as energy-efficiently as possible. Appropriate distribution of tasks among devices might be a basic approach leading to obtaining energy savings [56]. Especially for different GPU cards, the analysis and prediction of performance and energy consumption can be used to create a new policy and apply it to tools such as SLURM [57]. Another functionality that a suitable scheduler can include is the consolidation of GPU tasks [58]. If it is possible to perform two tasks simultaneously, this can bring benefits in terms of efficiency and energy savings. The tasks must be linked in such a way that the memory and the number of SMs are sufficient for execution. For additional efficiency and energy optimization, additional auto-tuning tools may be used to adjust the number of blocks, the number of threads per block, and the frequency (allowed by modern GPU cards) [59]. Using a tool such as NVIDIA System Management Interface (nvidia-smi) for power capping as well as in the case of a CPU can create additional configurations [49]. In addition, schedulers can be adapted to specific tasks. For machine learning (ML) applications, a system with dynamic batching and coordinating DVFS of the GPU was effectively used to reduce power consumption [60]. Energy-aware schedulers are also widely studied in applications such as GPU-RAID [61,62]. In terms of scheduling, a programmer should also keep in mind that the use of the CPU has an impact on the performance of the GPU, which is used for communication, among others [63]. In this case, if the scheduler running on the CPU absorbs too many resources, it can affect the work of the GPU.

3.3.3. Heterogeneous Environments

Currently, the heterogeneity of high-performance computing systems is present both within individual computing nodes, due to the presence of multicore CPU(s) and GPUs or other types of accelerators, and when integrating many non-homogeneous nodes, either within LAN or cluster environments. Some models (later also called heterogeneous processors) specify the usage of heterogeneous compute devices (can be multicore CPUs, GPUs, or other accelerators) that can be applied to either single-node or multi-node systems.

Due to the diversity of architectures in a heterogeneous system, more complex schedulers are required. It should be assumed whether jobs should be limited to a given pool of devices (due to the lack of implementation) or whether implementation should be enforced in frameworks that support cross-platform, such as OpenCL [64], Vulkan, and OneAPI. Additionally, power consumption and execution time prediction algorithms are required for different architectures because their power consumption and performance per task are different [65]. In heterogeneous environments, special attention is paid to communication due to possible bottlenecks that the scheduler must take into account. This is also an important aspect because, for some low-complexity jobs, the turnaround time including communication time for the fastest devices (usually the GPU) may be longer than the execution on the CPU. The scheduler must also take into account that some of the resources are used for themselves (prediction, job scheduling, using energy-aware tools). When using mechanisms to control energy consumption, such as DVFS or power capping, it is necessary to cooperate with multiple tools [40]. In the case of a cluster CPU + GPU system, it is necessary to account for both CPU-GPU communication overhead and inter-node communication on the CPU side i.e., CPU cores needed for such purposes as not doing so will slow down overall processing [29,66].

3.4. Time and Power/Energy Measurements

Optimization using the aforementioned goals that involve metrics such as execution time and power as well as energy consumption requires proper measurement techniques.

Execution time is typically measured on the CPU side as it allows measuring the wall times between the start and end of computations spawned either on CPU cores or on accelerators such as GPUs the latter typically involving GPU management threads running on CPU cores [66]. Appropriate system calls such as `clock_gettime()` (clock_getres() acquires precision of a given clock clockid, potentially down to nanoseconds) or `MPI_Wtime()`

(intended to be a high-resolution, wall time clock function, in case of a parallel application running on a cluster) or similar are used.

In terms of power and energy measurements, either power measurements from APIs made available by computing device (CPUs and GPUs) manufacturers are taken or power readings from hardware meters (typically amounting to whole computing nodes) are used. For the former, often used APIs include Intel RAPL for Intel CPUs (also through interfaces such as PAPI) or NVIDIA NVML for NVIDIA GPUs. We shall note that NVIDIA declares NVML power readings being accurate within ±5% of the current power draw. The `nvmlDeviceGetPowerUsage()` function can be called to retrieve the power usage of a device—in paper [67] the sampling frequency of reading the sensors was set to a maximum of 66.7 Hz. In the same paper, the authors performed measurements of energy consumption of approx. Forty instructions executed on Maxwell, Pascal, Volta, and Turing NVIDIA GPUs and verified that for all tested instructions average mean absolute percentage error (MAPE) of the MTSM (multi-threaded synchronized monitoring) software method (using readings from NVML) turned out to be 6.39 while mean root mean square error (RMSE) was equal to 3.97. Typically used hardware meters in this context belong to the Yokogawa WT300 series power meter with basic accuracy of 0.1% and 100 ms data update rate. Energy measurements are obtained by integrating power data over time.

Some works have reported inaccuracies of APIs such as Intel RAPL and NVIDIA NVML compared to hardware meters and consequently, unsuitability of incorporating the former into mechanisms for dynamic energy-aware optimization [68]. On the other hand, a comparison of the accuracy of RAPL and NVML against hardware measurements has been found adequate for power-capped optimization of energy, EDP, and EDS for CPUs in [10], where power caps were applied for application runs with specific problem sizes.

4. Energy-Aware Scheduling Algorithms

In this section, selected energy-aware scheduling algorithms are described and classified by their types. Their most important features are summarized in Table 2 and concisely visualized in Figure 1.

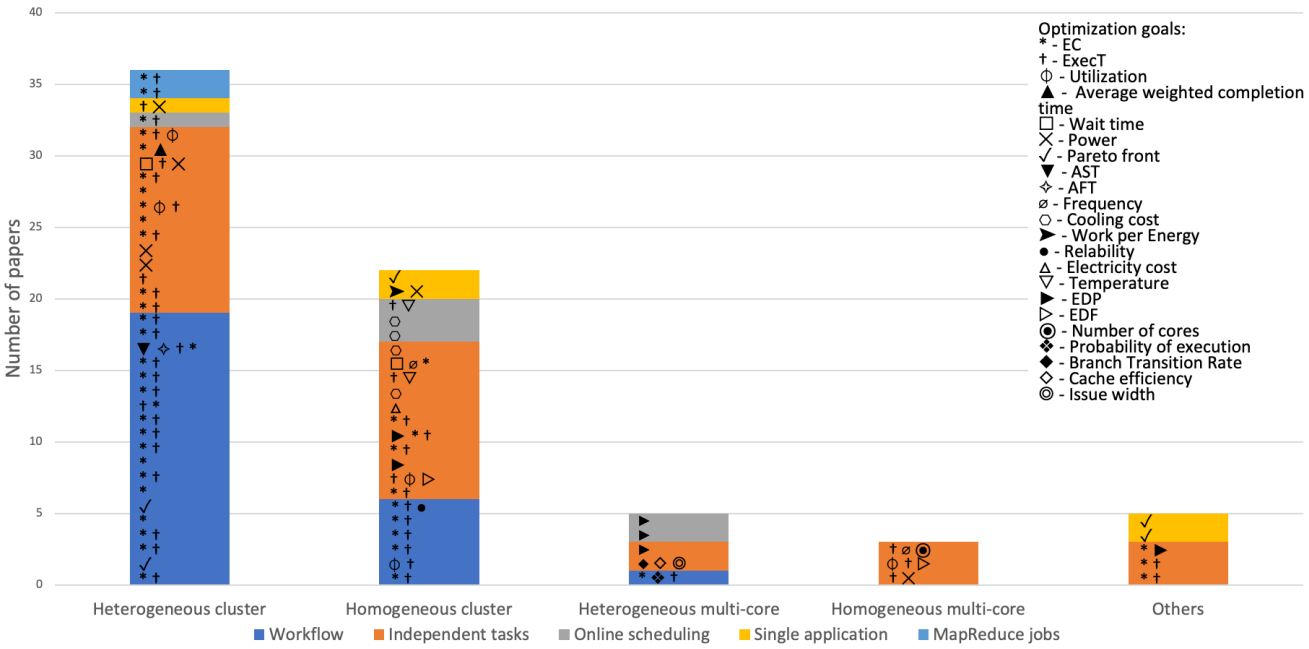

Figure 1. Analyzed papers with scheduling algorithms.

4.1. Randomized Algorithms

In [69], the authors first studied CPU-GPU utilization models obtaining power-CPU utilization and power-GPU utilization dependencies and focused on optimization of an

energy-efficient job scheduling problem with conditions on both the deadline and utilization (as close to optimal as possible) constraints and minimizing job execution energy consumption. Then they compared a Utilization Aware and Energy-Efficient Heuristic Greedy Algorithm (UEJS) and a Hybrid PSO Algorithm (H-PSO) showing benefits of the latter in terms of average energy consumption compared to UEJS by 36.5%, Max-EAMin by 36.3% and GA by 46.7%.

In [70], minimization of EDP formulated as a product of mean stretch time and normalized mean power is considered for job scheduling on the condition that no two jobs are scheduled on a server at the same time. An energy-aware greedy algorithm with particle swarm-optimized parameters is proposed and it was shown that a solution can be optimized by using malleable jobs (the number of allocated servers can be changed at runtime) and powering off idle servers.

In [71], the authors tackled the problem of finding a non-preemptive schedule of a set of jobs minimizing the average sum of energy and weighted completion time and proposed a randomized approximation algorithm that they subsequently de-randomized in order to find an approximation algorithm that was deterministic. This is considered in the context of scheduling on unrelated parallel machines taking into account time and energy as well as release dates depending on machines. Subsequently, they extended their approach to an algorithm minimizing average completion time with a bound on energy consumption.

4.2. Machine Learning

In [72], the authors proposed a framework for data centers that allows obtaining interesting energy consumption and response times for scenarios involving incoming job streams without any stationary assumptions regarding jobs. The approach is based on reinforcement learning (RL) through constructing a proper reward—negative of the weighted sum of average job response time and average power consumption within a time slot. The authors used Google cluster usage data trace and were able to obtain energy savings of 5.4%, 17.8%, and 24.5% at the cost of 0%, 29%, and 73% extensions of the job response times, respectively.

In [46], the authors proposed to use two mechanisms for the reduction of power consumption in a data center while preserving an SLA. This involved turning off machines that are idle and consolidating, i.e., minimizing the number of machines used. For the latter, they proposed to use supervised machine learning for the prediction of power consumption and user satisfaction level of a job. This is then considered a machine learning scheme using a dynamic backfilling scheduler. Various turnoff and -on thresholds were considered and power vs. SLA fulfillment has been analyzed. Finally, the authors compared five scheduling algorithms i.e., random, round robin, backfilling, dynamic backfilling, and their machine learning DB, for three workloads: grid (obtained for Grid5000), service (from Ask.com), and heterogeneous (mix suited for data centers). While for the grid workflow, original backfilling algorithms were considerably better than the others in terms of power consumption, for both the service and the heterogeneous workloads the proposed algorithm offered considerably better power consumption by approx. 16% and 7% at the slight CPU usage penalties of 3% and 4%, respectively.

In [73], the authors proposed an energy-aware partitioned-based task scheduling algorithm using deep reinforcement learning which firstly places tasks in partitions and, subsequently, tasks are assigned servers within those partitions. They proposed an energy consumption model and the usage of an autoencoder to deal with a multi-dimensional model to speed up convergence. The solution was tested in small, medium, and large scale environments showing generally superior results compared to Tetris and H20 in terms of normalized accumulated energy consumption and in between (larger than Tetris) normalized accumulated waiting time, e.g., for 120,000 tasks saving approx. 27.5% more energy compared to Tetris and increasing the waiting time by 6.8%.

In [74], the authors have adopted machine learning in order to optimize energy efficiency defined as the amount of work completed per unit of energy and learning

preferred socket allocation, using HyperThreading and CPU DVFS settings in order to optimize energy efficiency at various points of application execution. This approach does not require any code changes and maps a selection of hardware counter values onto the aforementioned settings but may require costly training. The authors used an 80 physical core (16 logical core) system with 512 GB DRAM and 4 sockets as well as 21 benchmarks for training, including a selection of, in particular, NAS Parallel Benchmarks, AMG, Kripke, LULESH, Quicksilver, XSBench, RSBench, CoMD, HPGMG-FV [2], Jacobi solver, and STREAM. Results were evaluated using bioinformatic HPC applications HipMer, IDBA, Megahit, and metaSPAdes, reaching a 39% reduction of power consumption with an average increase of execution time of 31%.

In [49], the authors considered the problem of scheduling jobs being HPC applications onto a parallel system meeting a power bound optimizing bounded slowdown (BSLD) which is the ratio of actual wait and execution time divided by specified job runtime. The approach uses a machine learning model for predicting CPU power consumption (the authors simplify the GPU estimation by adding maximum power consumption of GPUs if engaged) and scheduling either using the priority rules-based (PRB) algorithm or a hybrid solution with a relaxed constraint programming (CP) model for an initial schedule and a heuristic approach for the final schedule. They have reported an average gain of 8.5% vs. state-of-the-art testing for the Eurora supercomputer.

4.3. Dynamic Programming

Dynamic programming [75,76] is used for energy-aware scheduling for a heterogeneous multicore processor architecture in [77]. Characteristics of this work are assumptions that consider a workflow as well as limiting execution time and assuming the probability of execution while minimizing energy. The authors have proposed a minimum energy under probability constraints algorithm to select a core and voltage level for each task to satisfy the probability constraints, subsequently, a leaf-partition algorithm is applied for finding an execution order and a trading energy for time algorithm is used for minimization of execution time. The solutions were shown to demonstrate results better than existing algorithms with up to approx. 30% improvement.

4.4. Fuzzy Logic

In [78], the authors adopt the fuzzy logic [79] approach for scheduling programs to appropriate cores within a system—based on metrics such as instruction dependency distance, data reuse distance, and branch transition rate. The proposed scheduling approach has been shown to provide up to 15% average reduction in EDP compared to random scheduling. The authors tested a single ISA heterogeneous four-core processor with various specifications for each core and integer and floating point SPEC CPU2000 benchmarks.

4.5. Integer Programming

In the first phase of the approach proposed in [80] jobs are scheduled with heuristics based on a greedy algorithm or integer linear programming (ILP/LP)—the HILP model (several metrics based on utilization, execution time, and earliest deadline). In the second phase, frequency is adjusted using DVFS and the designed integer linear programming (ILP) model. Tests on synthetic data showed a large acceleration of HILP over typical ILP in finding the right solution to the proposed model. The algorithm also can select hyperparameters to establish a tradeoff between performance and energy.

In [81], the authors tackled the problem of static scheduling of moldable (i.e., with a fixed degree of parallelism) tasks with a focus on energy minimization while maintaining time deadlines. They test how core allocation, assignment, frequency scaling, and ordering, subject to the application of constraints, impact the quality of the solution. Experiments were performed for synthetic task sets and StreamIt benchmark suite applications. A generic core with a power consumption model analogous to ARM's big.LITTLE architecture was used. The authors concluded that except for frequency constraints, most constraints do

not allow for lowering scheduling times compared to a solution with an ILP. They also considered stepwise scheduling with the steps allocation, mapping and ordering, and frequency scaling and assessed heuristics for the steps compared to ILP. A fully heuristic solution was inferior to ILP for small and medium size tasks, they also indicated the possibility of quality heuristics for frequency scaling, which not being the case for allocation and mapping.

In [82], the authors proposed MaxJobPerf, which is an ILP-based scheduling algorithm for a job scheduling problem with two constraints: processors as well as available power. The objective function maximizes longer waiting jobs and higher frequencies deciding which jobs from the current waiting queue shall be launched and run at what frequencies. They compared the new policy against EASY without frequency scaling and PB guided showing better results in the context of the bounded slowdown (BSLD) metric. The authors used an Alvio scheduling simulator with a DVFS feature and the proposed MaxJobPerf.

4.6. Evolutionary Algorithms

Maximizing peak inlet temperature within a data center via task assignment has been tackled in [83]. Two solutions were proposed to solve the problem: XInt-GA using a genetic algorithm as well as XInt-SQP adopting quadratic programming (firstly used a solution using a real domain and finding a close integer solution meeting constraints). In terms of results performed for a simulation of a data center, the authors argue that the reached inlet temperatures, compared to existing approaches, are 2–5 Celsius degrees lower and ultimately allow for 20 to 30% cooling cost reduction.

A typical example of an evolutionary solution for the energy-aware scheduling problem was presented in [84]. The authors used the workflow version of the problem along with a specific structure of the destination system represented by an undirected graph of compute nodes, connected by edges related to network links. The authors presented three algorithms: (i) plain genetic (Plain GA), (ii) cellular automata supported by genetic approach (CA + GA), and (iii) heuristic, giving preferences to high-efficiency machines in allocation (EAH), based on a typical FIFO algorithm. The used fitness function is based on energy minimization only. The simulation experiments showed the new algorithms having similar performance results in comparison to typical, time-only approaches, however, the energy consumption seemed to be much lower. The best results were achieved by CA + GA; however, other algorithms were much faster and more memory efficient, especially EAH.

In [85], Mezmaz et al. proposed a new parallel bi-objective hybrid heuristic algorithm combining a genetic approach with an earlier released greedy solution [86] adapted to validate and evaluate the offspring created with crossover and mutation operations. They used a heterogeneous environment (different compute node types) and dynamic voltage scaling (DVS) to adjust the power level applied to an application, represented as a workflow. The algorithm returns a whole set of results, being the approximation of the Pareto dominant solutions (front), related to different energy-performance pairs; thus, no specific function/metric is used. The performed experiments, based on an FFT real-world application, showed a reduction of energy consumption by 47.5% with the 12% completion (execution) time penalty.

In [87], Guzek et al. proposed three evolutionary scheduling solutions, based on well-known multi-objective evolutionary algorithms: NSGA-II [88], MOCell [89], and IBEA [90]. As in the previously described approach, the scheduled tasks are organized into a workflow, for the power differentiation the environmental heterogeneity and DVFS technique were exploited, also the result was presented as a whole collection of the (possibly) Pareto-dominated solutions, thus no specific metric was used. After the performed analysis, all three algorithms proved to converge to accurate solutions and confirm their effectiveness. No real-world application tests were performed, even in a simulation environment.

Two new GA-based algorithms for scheduling a batch of independent tasks in a computational grid environment were described by Kolodziej et al. in [91]. The proposed

solution was based on a hierarchical approach, with performance being the prioritized objective, so no metric was necessary. The power was controlled using various types of compute nodes supporting DVFS technology. The solution was tested using a HyperSim-G simulator, with different sizes of task batches and computational grids. The results showed a possibility of significant energy reduction, up to 33% savings, but with no comparison to other, pure performance-oriented, algorithms.

In [92], Kassab et al. assessed four genetic algorithms for scheduling a bag of tasks with a global power cap set for cores of one CPU, keeping the objective of the optimization to minimize the execution time. The energy consumption is controlled by selecting specific tasks to execution, while they differ in peak power, provided before the scheduling. The proposed algorithms differ in their approach to performing crossover operations: 1pX-W—with only one point crossover, OX-W and MX-W—both with two-point crossover, the former with order crossover and the latter with the classical one, and finally noX-W—without crossover at all. All of them use wheel selection and are typical to genetic algorithms mutation operations. The experiments were performed using an eight-core CPU simulator, showing an advantage of the GA approach over list scheduling algorithms [93].

4.7. Constraint Programming

A new scheduling method, based on a queue system, similar to that used in HPC centers, e.g., SLURM (https://slurm.schedmd.com—SLURM: an open source, fault-tolerant, and highly scalable cluster management, and job scheduling system, accessed: 15 December 2022) was proposed by Borghesi et al. in [49]. The authors used constraint programming to impose a power limit on the whole HPC cluster, where the differences between the power consumption of the queued jobs enabled control of the total power usage. The simulation experiments were performed using historical traces of the Eurora supercomputer (468th on TOP500 in June 2013) and showed good results (better performance) in comparison with RAPL power capping and DVFS purely based methods.

4.8. Other Algorithms

Article [94] by Mishra et al. presented two approaches for real-time workflow scheduling. The both gave performance higher priority and focused on reclaiming slacks between tasks execution, using DVFS to lower faster tasks speed and decrease power usage of the whole execution. The first approach, static power management (S-PSM), shifted a static schedule reducing slacks related to the worst-case workflow execution, while the second, dynamic power management (DPM), additionally tried to reclaim the stack caused by tasks finished earlier than their worst-case scenario. The authors implemented several algorithms based on the aforementioned approaches and performed their tests in a simulator environment, showing up to 20% energy-saving improvement in comparison with other schedulers.

Chesie et al. proposed two algorithms based on well-known first in, first out (FIFO) and backfill (BFF) strategies, where an additional constraint related to peak power limit (cap) was introduced [95]. They enable the scheduling of GPU jobs from a queue, with defined maximum power consumption for each job; thus, no hardware power management mechanisms, such as DVFS, were necessary. The experiments showed peak energy reduction up to 24% and the execution time increased by around 2%.

In [96], Wang et al. presented a workflow scheduling algorithm for a homogeneous cluster, with a possible tradeoff between energy and execution time, where the user defines an additional constraint: a maximal time of delay in percent; thus, no additional metric is needed. The solution is based on the earliest task first (ETF) approach, with scaling down frequencies (using DVFS) of the non-critical tasks, and in case of indicated tradeoff, also for the critical ones. The experiments performed in the simulation environment showed 44% of the maximal energy reduction and the solution superiority over other energy-aware algorithms in this aspect.

In [97], Nesmachnow et al. proposed a collection of 20 energy-aware heuristics for a heterogeneous grid environment, based on a list scheduling algorithm, where power control is imposed by differences in task execution on various compute nodes, without using DVFS or other direct hardware mechanisms. The proposed solutions differ in a degree of tradeoff, which varies from performance only, through percentage-based best matching of task-machine pairs, to pure energy approach; thus, no fitness function (metric) is defined. The authors argued the proposed heuristics are more appropriate than other existing aggregating and Pareto approaches, showing the results of experiments performed in a simulation environment.

Article [98] by Aupy et al. described several heuristics for scheduling a workflow by setting up the maximum execution time constraint but also taking into account the third objective: reliability. The proposed heuristics are dedicated to a continuous model of processor speed, where its frequency can be set up freely (note, that a typical DVFS technology uses a discrete approach). The simulation results showed that using frequency scaling and task re-execution mechanisms, it is possible to increase reliability while reducing energy consumption.

Rizvandi et al. in [16] proposed an algorithm (MMF-DVFS) for reclaiming slacks in a previously prepared performance-only schedule, e.g., list scheduling, where only two (minimal and maximal) DVFS frequencies are used. The authors provide the simulation results showing that their approach has higher energy reduction, without a significant performance influence than the reference algorithm (RDVFS), based on the whole range of frequencies.

In [99], Yang et al. proposed an algorithm scheduling queued jobs (sets of tasks) into a homogeneous cluster of compute nodes. Their goal was to decrease the power consumption during on-peak hours when the cost of energy is lower; thus, accumulating the power-intensive jobs during off-peak hours. In general, the utilized energy amount is the same, and the goal is to decrease the total cost, as is presented in the electricity bill. Their solution introduced the 0-1 knapsack policy, which selects optimal (most or least energy consuming, for off-peak and on-peak hours, respectively) jobs within a defined time window. The simulation experiments, based on real cluster (IBM Blue Gene/P) traces, showed a cost reduction of up to 23%.

In [100], Barik et al. presented algorithm partitioning computations between CPU and GPU, using developed power models of hardware and software components, implemented for a tablet and a workstation with integrated CPU/GPU chips. The scheduling algorithm performed a linear search for the best solution, using the sixth-order polynomial approximation of the energy-performance hardware model and online profiling for the workloads, which need to be adapted for flexible CPU and GPU execution. The experiments performed for EDP and pure energy objectives showed results very close (over 90%) to the ones delivered by a near-perfect brute force solution.

An energy-aware task scheduling algorithm (EAMM) algorithm based on the Min-Min method [101] was presented in [102]. It is a batch schedule of a set of independent tasks that prioritizes the shortest expected job completion time (calculated taking into account the power state). The algorithm was designed for batch-type scheduling. EAMM can save 34% of energy compared to the base Min-Min version without extending the execution time (up to 1%).

A different approach of batch scheduling was proposed in [103]—the min-energy-max-execution (MEME) algorithm. It takes turns assigning jobs with the lowest energy consumption and the highest execution time. The authors showed the advantage of the algorithm in execution time and energy savings over Min-Min, MCT [101], and PPIA [104] scheduling algorithms.

Work [105] describes energy-aware scheduling by minimizing duplication (EAMD). This is a duplication-based scheduling modification that reduces energy consumption by minimizing the number of duplications. Compared to other duplication-based algorithms, heterogeneous limited duplication (HLD) [106] and heterogeneous critical parents with fast

duplicator (HCPFD) [107], they gain energy savings of up to 15.59%. The algorithm can also be used as an additional phase to other scheduling algorithms.

The authors of [108] proposed two algorithms, energy-aware duplication (EAD) and performance-energy balanced duplication (PEBD) based on duplication. These algorithms add duplication but take into account the use of energy. PEBD differs from EAD in that it compromises energy consumption and performance. The results show an improvement in a performance-energy tradeoff of up to 20% over other traditional duplicating and non duplicating algorithms.

Article [109] proposes the energy-efficiency with duplication (EED) and energy-efficiency with non-duplication (EEND) heuristics. The algorithms work under the constraints of execution time. EED strives to reduce duplication and pool resources with the subset relationship. EEND is based on EED scheduling but omits duplication execution. The algorithm gains 50% energy savings over the EAD algorithm. EED can improve the normalized execution time by 18% and the EEND has a worse normalized execution time by 25% to the TDS reference [110].

In [111], Mammela et al. used dynamic power management (DPM) mechanisms, including switching off the entire compute nodes, with the the well-known scheduling algorithms first in, first out (FIFO), backfill first fit (BFF), and backfill best fit (BBF), turning them into their energy-aware versions: E-FIFO, E-BFF, and E-BBF, respectively. The new algorithms were compared to the base ones, obtaining 6–16% energy savings (the largest savings for E-FIFO), in simulation and real testbed HPC environments with relatively low resulting delays for both execution time and task response time: up to 3.2% and 2.3%, respectively.

For map-reduce jobs, the energy-aware scheduling EMRSA algorithm and its modifications EMRSA-I and EMRSA-II were proposed [112,113]. The algorithms work using a queue, taking into account their defined metric—energy consumption rate, time of map, and reduce jobs as well as deadlines. Tests on the Hadoop cluster show that EMRSA, EMRSA-I, and EMRSA-II minimize an average of 40% of energy than known execution time minimization algorithms.

Article [114] describes an algorithm called EDL for offline (batch) and online scheduling using DVFS. The algorithm dynamically sets the value of DVFS parameters and uses its heuristic for scheduling jobs. A heuristic is based on the model built and uses, among others, the earliest deadline first (EDF) metric. The algorithm can achieve 30–33% average energy savings, almost reaching the theoretical limit of the model, which is 35%. The comparison was made by real-world power measurement traces.

Expanding the EAS, the Linux job scheduling algorithm was proposed in [115]. Scheduling takes into account execution time and energy consumption based on previous runs. Using NAS Parallel Benchmarks [116], the authors show possible tradeoffs between energy and execution time. For selected test cases, the algorithm can save up to 21.5% of energy at the expense of 3.8% of execution time.

The energy-aware service level agreement (EASLA) [117] algorithm adjusts the maximum slack to the maximum subsets of independent jobs to reduce energy consumption and improve parallel jobs execution. The test algorithm achieved from 10.49% to 22.68% compared to slack distribution approaches. Improvement of the sub-algorithm for EASLA was proposed in [118]. The authors used DVFS to downscale frequency and ensure the same execution time by predicting schedule length. The upgrade can save up to 14.93% more energy compared to the base version of EASLA.

In [119], energy-aware dag scheduling (EADAGS) was proposed. The scheduler is based on the use of DVFS and decisive path scheduling (DPS) [120]. After scheduling with DPS is done, DVFS is used during slack times. This solution gained a 40% energy-saving advantage with the same execution time over the base DPS.

The energy-aware forward list scheduling (eFLS) algorithm [121] sorts jobs by worst-case execution time (WCET) using depth-first search (DFS) and breadth-first search (BFS) taking into account the use of DVFS. Dedicated tie-breaking methods (time laxity, energy

laxity) were used to generate multiple schedules and choose the one with the lowest energy consumption. The authors managed to obtain a heuristic almost consistent with the presented ILP model (deviation up to 15.8%). The heuristic saves an average of 25.1% more than the base forward list scheduling (FLS) algorithm but with an average of 19.3% longer execution time.

In [122], virtualized homogeneous earliest start time (VHEST) and the energy-aware scheduling algorithm (EASA) were proposed for virtualized data centers. The first VHEST algorithm is based on the well-known HEFT algorithm [123] with an overlapping insertion policy. EASA is a multi-objective heuristic to maximize utilization and minimize execution time. The authors demonstrated a significant advantage of EASA compared to VHEST and HEFT in terms of energy saving. EASA saved an average of 31% more energy with less than 2% longer execution time than HEFT (EASA is 18% better than VHEST in terms of energy savings).

The list scheduling algorithm and variations were proposed in [124] (the algorithm is called LS for CPUs) and [125] (the algorithm is called LESA for heterogeneous). In both cases, a series of metrics to sort the list are described. The LS authors have proposed many modifications to the algorithm that outperforms the LS algorithm in the defined validation metric—normalized schedule length (NSL). The LESA algorithm also imposes energy usage limits on jobs using DVFS and the computed limits based on the model. Under energy constraints, compared to other scheduling algorithms, including the well-known HEFT, LESA gets an average of 1.81 times smaller execution time.

Energy-dynamic level scheduling (EDLS) [126] adjusts energy utilization with DVFS to obtain energy-efficiency tradeoffs for a workflow of jobs. EDLS is an energy-aware modification of the DLS scheduling algorithm [127]. It prioritizes jobs that require less energy. For the algorithm, the energy dynamic level (EDL) metric is introduced, taking into account the energy use in the dynamic level (DL) metric described in [127]. The algorithm saved up to 70% of energy compared to the EDL base algorithm.

Spatio-temporal thermal-aware online scheduling. described in the article [128] is based on thermal constraints, which are provided by adjusting the DVFS parameters. Scheduling also considers minimizing the execution time by matching jobs to resources to balance load times. The authors compared their solutions to round robin and assigned jobs to resources with the lowest temperature. The proposed algorithms gained an advantage in minimizing the execution time by up to 10% while reducing energy consumption.

Article [129] describes the power-aware algorithm for scheduling (PAAS) using a knowledge base to regulate DVFS minimizing energy consumption. The job assignment itself is done taking into account the shortest processing time. The algorithm reduces energy consumption by an average of 12.64% comparing the optimal frequency and voltage to the maximum while ensuring optimal execution time.

The reformed scheduling method with energy consumption constraint (RSMECC) [130] calculates the energy consumption for the entire workflow and DVFS energy constraints for each job based on the defined model. The algorithm generates a schedule for each assignment and chooses the best one based on the defined job start times, jobs finish times, energy consumption, and execution time. The algorithm was compared with, inter alia, the HEFT algorithm, yielding a shorter execution time for the given energy constraints.

The energy-conscious scheduling (ECS) heuristic described in [86] makes a trade-off between efficiency and energy consumption by minimizing the bi-objective function through scheduling decisions. ECS also has a second phase in which it analyzes possible energy savings without increasing the execution time. In addition, a modification to the ECS + idle [131] algorithm was proposed, which takes into account the energy consumed during the idle state. The algorithm was compared with the aforementioned HEFT and duplication-based bottom-up scheduling (DBUS) [132], which do not take energy into account. The algorithm uses DVFS. The algorithm improves power consumption up to 12% vs. HEFT and up to 46% vs. DBUS, execution time up to 5% vs. HEFT, and up to 23% vs.

DBUS. ECS-idle improved execution time and energy consumption by an average of 2% and 7% compared to ECS.

Article [133] describes an energy-aware scheduling algorithm for parallel application (ESPA) using DVFS and DPM techniques. The algorithm clusters jobs to select an appropriate schedule. Then it selects energy management strategies based on the type of jobs (critical, communication time, idle time). The algorithm can save 12.95% more energy than HEFTA and shorten execution time by 2.11%.

The algorithm proposed in [134] is a Quantum-Inspired Algorithm (QHA) for designing and managing appropriate heuristics (hyper-heuristic) for energy-aware scheduling. Choosing a heuristic allows getting better results for different types of applications and configurations. The algorithm can speed up the heuristic search process by 38 percent to the standard search method. QHA lowered the energy consumption ratio by 26.5% and 11.3% over HEFT and ECS.

In [135], the Cuckoo search algorithm (GACSM) using Gaussian random walk to balance exploration and exploitation and adaptive discovery probability modifications for greater population diversity was used for the problem of scheduling and matching DVFS values. The algorithm applies a final heuristic to improve performance based on a defined cost-to-time metric. Energy savings were 14.9% greater than EASLA, 6.9% greater than ICMPACO, and 8.4% greater than QHA.

The energy-aware stochastic task scheduling algorithm (ESTS) proposed in the article [136] is a heuristic that minimizes execution time and energy consumption at the same time. The algorithm has $O(|J| * (|R| + log|J|))$ complexity. Developers create heuristics based on their metrics of the probability of meeting the constraints and utilize DVFS. Energy consumption and execution time experiments have shown the advantage of ESTS over reference EDF-DVFS [137] by 20.3/4.2% and 2.4/18.8% (depending on the input parameter to the algorithm).

In [138], through the proposition of a regression model for estimating energy consumption on a heterogeneous system, the authors proposed mapping a program onto an appropriate core in various phases. They demonstrated that the proposed approach can give approx. 10–20% EDP reduction compared to static and periodic sampling approaches, respectively, tested on the Intel QuickIA platform for astar, bzip2, h264ref, hmmer, lbm, and libquantum and using SPEC CPU2006 and denoise, the deblure, reg, and seg benchmarks.

In [139], the authors deal with the optimization of EDP for applications running on heterogeneous multicore processors. The approach assumes that various phases of an application are identified at runtime and for each thread-to-core (assuming heterogeneous cores) is found and applied. The proposed phase EDP has been evaluated in a simulator showing a 16% average and up to 29% reduction in EDP compared to all possible static assignments with a minimal to moderate reduction in speed-up when optimizing energy consumption.

In [28], the authors presented an approach including an algorithm and a solution for the selection of computing devices such as CPUs and GPUs as well as subsequently partitioning and assignment of data packets to these devices in a cluster, to be processed by OpenCL kernels, also taking into account communication costs such that total execution time is minimized while the power consumption of selected devices under load is below a predefined threshold. As a solution, a greedy approximation algorithm was applied to solve the optimization problem generalized to a 0/1 knapsack formulation. It was also shown how execution time depends on the number of data packets generated for a particular power limit as too small a number results in the inability to balance the load and too large causes overhead due to communication (specifically due to communication startup times) and management costs. For an application of MD5 password breaking and a heterogeneous cluster with Core i7, GTS, two GTX as well as Tesla and Quadro GPUs, for a range of power limits from approx. 200 to 1500 W, measured speed-ups almost match simulated ones considering scheduling and communication overheads and consistently increase with increasing power limits.

In [140], the authors consider optimization towards obtaining Pareto fronts that involve execution time and energy consumption of a workload executed on a homogeneous cluster with multicore CPUs. Firstly, the authors demonstrate that for homogeneous clusters with multicore CPUs, because of NUMA and resource contention, dependencies of speed (MFlop/s) and dynamic energy consumption are not smooth and non-linear versus problem size (such as for FFTW executed using 24 threads on Intel Haswell CPUs). Then they formulate a bi-objective optimization problem considering performance and energy consumption BOPPE that takes as input processor speed and energy profiles versus problem size. As a solution to the problem, the authors proposed the ALEPH algorithm, which determines globally Pareto-optimal solutions for energy and performance of a workload of a given size (certain granularity is assumed) of complexity $O(m^2p^2)$ (p—number of processors, m—cardinality of sets representing speeds and energy values). For matrix multiplication and FFT applications and assumed 5% performance drop, they saved on average and maximum 9/44% and 8/20% respectively. Additionally, the authors showed the possible coupling of ALEPH with DVFS for a better set of Pareto solutions. In [141], the authors presented HEPOPTA for data partitioning for bi-objective optimization considering execution time and energy for data-parallel applications executed in a system that consists of several heterogeneous processors. The algorithm takes as input discrete dynamic energy and time functions (vs. data size) and obtains Pareto fronts for imbalanced solutions. The authors benchmarked applications such as matrix multiplication, 2D fast Fourier transform, and gene sequencing using two connected heterogeneous servers with CPUs, GPUs, and Intel Xeon Phi, and demonstrated significant gains in execution time and dynamic energy used compared to balanced solutions e.g., on average 26% and 130% for matrix multiplication, 7% and 44% for FFT, and 2.5% and 64% for gene sequencing. In [142], the authors applied an algorithm for a continuous case (execution time strictly increasing and energy linear increasing) for optimization of parallel application execution on a hybrid heterogeneous platform to two problems: optimizing for dynamic energy and performance as well as for total energy and performance. They showed that a given solution vector is Pareto-optimal for execution time and total energy if and only if it is Pareto-optimal for execution time and dynamic energy. They presented maximum total energy savings of 8% (performance drop pf 5%) for matrix multiplication and 16% (performance drop pf 1%) for gene sequencing, using a system with two Intel CPUs, NVIDIA K40C P100 and an Intel Xeon Phi.

Table 2. Scheduling algorithms, systems, and optimization metrics used for them.

Algorithm Type	Algorithm	Optimization Goals/Metrics (O—Optimized, C—Constrained)	System Type	Energy-Aware Mechanism	Workload Type
Randomized algorithms	UEJS with H-PSO [69]	EC (O), ExecT (C), Utilization (C)	Heterogeneous cluster		Independent tasks
	Particle Swarm optimized greedy algorithm [70]	EDP (O)	Homogeneous cluster		Independent tasks
	LP relaxation [71]	Average weighted completion time (O), EC (O, C)	Heterogeneous cluster		Independent tasks
Machine learning	RL based scheduler [72]	EC (O), Weighted ExecT (O)	Homogeneous cluster		Independent tasks
	ML approach based on supervised learning [46]	EC (O), Machines usage (O), ExecT (C)	Heterogeneous cluster		Independent tasks
	DRL [73]	EC (O)	Heterogeneous cluster		Independent tasks
	ML Classifiers [74]	Energy efficiency (amount of work completed per unit of energy) (O), Power (C)	Homogeneous cluster	DVFS	Single application
	Scheduling-based Power Capping using CP and ML [49]	Wait time (O), ExecT (O), Power (C)	Heterogeneous cluster		Independent tasks
Dynamic programming	Accelerated Search [77]	EC (O), Probability of execution (C), ExecT (C)	Heterogeneous multicore	DVFS	Workflow
Fuzzy logic	Important inherent program analysis [78]	Branch Transition Rate (O), Cache efficiency (O), Issue width (O)	Heterogeneous multicore		Independent tasks
Integer programming	Search space design for search unrestricted, crown, bookshelf, and pipe schedulers [81]	Complex objective function (ExecT, number of cores, frequency) (O), ExecT (C)	Homogeneous multicore	DVFS	Independent tasks
	MaxJobPerf [82]	Wait time (O), Frequency (O), EC (C)	Homogeneous cluster	DVFS	Independent tasks
	XInt-SQP [83]	Cooling cost (O, C)	Homogeneous cluster		Independent tasks, Online scheduling
	HILP [80]	Utilization (O), ExecT (O, C), EDF (O)	Homogeneous multicore/cluster	DVFS	Independent tasks
	RNRA, RIRA [143]	EC (O)	Heterogeneous cluster	DVFS	Independent tasks
Evolutionary algorithms	XInt-GA [83]	Cooling cost (O, C)	Homogeneous cluster		Independent tasks, Online scheduling
	Plain GA [84]	EC (O)	Heterogeneous cluster		Workflow
	Plain GA, CA + GA [84]	EC (O)	Heterogeneous cluster		Workflow
	Parallel bi-objective hybrid genetic algorithm. [85]	Pareto front (O)	Heterogeneous virtualized cluster	DVFS	Workflow
	NSGA-II, MOCell, IBEA [87]	Pareto front (O)	Heterogeneous cluster	DVFS	Workflow
	GA with elitist or struggle replacement mechanisms [91]	EC (O), ExecT (O)	Heterogeneous computational grid	DVFS	Independent tasks
	1pX-W, OX-W, MX-W, and noX-W GA with power constraints. [92]	ExecT (O), Power (C)	Homogeneous multicore		Independent tasks
Constraint programming	Hybrid dispatcher [49]	Power (C)	Heterogeneous cluster		Independent tasks
Other	Power-aware scheduler [95]	Power (C)	Heterogeneous cluster		Independent tasks
	EAH [84]	EC (O)	Heterogeneous cluster		Workflow
	S-PSM, DPM [94]	EC (O), ExecT (C)	Homogeneous cluster	DVFS	Workflow
	ETF [96]	EC (O), ExecT (C)	Homogeneous cluster	DVFS	Workflow
	20 algorithms based on list heuristics [97]	EC (O), ExecT (O)	Heterogeneous computational grid		Independent tasks
	Greedy algorithm for knapsack problem with power constraints [28]	ExecT (O), Power (C)	Heterogeneous cluster		Single application
	Heuristics with continuous frequency scaling [98]	EC (O), ExecT (O, C), Reliability (O, C)	Homogeneous cluster	DVFS	Workflow
	Greedy policy, 0-1 knapsack policy [99]	Electricity cost (bills) (O)	Homogeneous cluster		Independent tasks
	Prediction and planning with a regression model. [138]	EDP (O)	Heterogeneous multicore		Online scheduling
	PRB [49]	ExecT (O)	Heterogeneous cluster		Independent tasks
	E-FIFO, E-BFF, E-BBF [111]	EC (O), ExecT (C)	Homogeneous cluster	DPM	Independent tasks
	EMRSA, EMRSA-I, EMRSA-II [112,113]	EC (O), ExecT (C)	Heterogeneous cluster		MapReduce jobs
	EDL [114]	EC, ExecT (C)	Heterogeneous clusters	DVFS	Independent tasks, Online scheduling
	Extended EAS [115]	EC (O), ExecT (O)	Heterogeneous clusters		Independent tasks
	EAMM [102]	EC (O), ExecT (O)	Heterogeneous cluster		Independent tasks
	EAMD [105]	EC (O), ExecT (O)	Heterogeneous cluster		Workflow
	CPU/GPU partitioning [100]	EC (O), EDP (O)	CPU + GPU node		Independent tasks
	MMF-DVFS [16]	ExecT (C), EC (O)	Heterogeneous cluster	DVFS	Workflow
	EASLA [117], Improved EASLA [118]	EC (O), ExecT (C)	Heterogeneous clusters	DVFS	Workflow
	QHA [134]	EC (O, C), ExecT (O, C)	Heterogeneous cluster	DVFS	Workflow
	EADAGS [119]	EC (O), ExecT (O)	Heterogeneous cluster	DVFS	Workflow
	eFLS [121]	EC (O), ExecT (C)	Heterogeneous cluster	DVFS	Workflow
	VHEST, EASA [122]	Utilization (O), ExecT (O)	Homogeneous virtualized cluster		Workflow
	EDLS [126]	EC (O), ExecT (O)	Heterogeneous cluster	DVFS	Workflow
	LESA [125]	ExecT (O), EC (C)	Heterogeneous cluster	DVFS	Workflow
	Spatio-temporal thermal-aware scheduling [128]	ExecT (O), Temperature (C)	Homogeneous cluster	DVFS	Independent tasks, Online scheduling
	PAAS [129]	EC (O), ExecT (O)	Homogeneous cluster	DVFS	Independent tasks
	EED, EEND [109]	EC (O), ExecT (C)	Homogeneous cluster	DVFS	Workflow
	RSMECC [130]	AST (O), AFT (O), ExecT (O), EC (C)	Heterogeneous cluster	DVFS	Workflow
	ECS [86], ECS + idle [131]	EC (O), ExecT (O)	Heterogeneous cluster	DVFS	Workflow
	ESPA [133]	EC (O), ExecT (O)	Heterogeneous cluster	DVFS, DPM	Workflow
	GACSM [135]	EC (O), ExecT (C)	Heterogeneous cluster	DVFS	Workflow
	LS [124]	EDP (O), EC (O, C), ExecT (O, C)	Homogeneous cluster		Independent tasks
	ESTS [136]	EC (O, C), ExecT (O, C)	Heterogeneous cluster	DVFS	Independent tasks
	EAD, PEBD [108]	EC (O), ExecT (O)	Homogeneous cluster		Workflow
	Phase_EDP [139]	EDP (O)	Heterogeneous multicore		Independent tasks, Online scheduling
	ALEPH [140]	Pareto front (O)	Homogeneous cluster	DVFS	Single application
	HEPOPTA [141]	Pareto front (O)	Heterogeneous processors		Single application
	LBOPA-TE [142]	Pareto front (O)	Heterogeneous processors		Single application

5. Conclusions, Open Problems, and Areas for Further Research

This article presents definitions for several versions of the problems (scheduling, resource allocation, workflow, and data partitioning). Optimization goals used in energy-aware HPC and techniques to reduce energy consumption are described. Modern scheduling algorithms for HPC systems, which are presented in Table 2, have been analyzed and categorized. We formulate conclusions based on the analysis of approaches, workloads, optimization goals, and target systems, described in previous sections and summarized in Figure 1:

1. We conclude that there is a variety of problem formulations and corresponding algorithm types that tackle the problem of energy-aware scheduling for HPC systems, including machine learning (with reinforcement and supervised learning), dynamic programming, fuzzy logic, integer programming, randomized algorithms, evolutionary algorithms, constraint programming, and others.
2. Most optimization goals involve metrics such as execution time/makespan, energy, and power, either in functions such as EDP or EDS or optimizing some while putting constraints on others, e.g., minimization of execution time under power limit. A limited number of works specifically consider cooling costs and temperatures.
3. Application models include mostly a bag or stream of incoming independent or periodic tasks or a workflow (DAG) composed of tasks in nodes of the graph with edges denoting dependencies.
4. System types targeted include mostly heterogeneous but also homogeneous clusters and homogeneous and heterogeneous multicore environments.
5. Most works use DFVS as a mechanism for controlling the power/energy of compute devices, some combine DVFS and DPM (including turning off machines), and a limited number of works use explicit power capping.

Additionally, based on the analysis of the area and specific works, we can further outline detailed problems and challenges that, we believe, require focus and solutions in the forthcoming period:

1. Which optimization goals shall be considered for what purposes e.g., consideration of EDP and EDS as ones involving time and energy (relative coefficients such as in EDS might depend on current electricity costs), pure energy or in some cases or areas minimization of execution time under power limit (across time) seems to become more important due to the risk of blackouts, etc.
2. Analysis of the impact of frequency of data monitoring on both accuracy (specifically referring to power/energy monitoring) as well as consideration of parameters applicable to data (power, load, etc.) filtering (such as low pass filters [9]) such as running averages which impacts the latency of energy-aware monitoring and correspondingly the scheduling algorithm vs. the possibility to deal with highly changing application and/or system load.
3. Following the discussion on measurement accuracy in Section 3.4, the accuracy of APIs such as Intel RAPL and NVIDIA NVML requires constant assessment in view of new generations of CPUs and GPUs and new APIs' versions. Additionally, another topic for investigation is use cases and conditions (possibly involving usage of data filtering) for which the aforementioned APIs provide reliable results compared to ground truth hardware meters.
4. Some works present other approaches to energy-/power-aware aspects of the computations, such as system efficiency or thermal-awareness in scheduling [14]. It is important to consider and analyze power under load for parallel applications by components such as fans and other cooling components, power supplies, and power distribution units (PDU) depending on their types and classes. Moreover, further consideration of temperatures, along with performance and power/energy consumption, in the context of power required for cooling/air conditioning can have a significant

influence on HPC systems. Specifically, power capping might affect not only execution times and power/application energy consumption but can lower the temperature.

5. The problem of finding (hyper)parameters for auto-tuning of algorithms for energy-aware scheduling. This might involve parameters of the scheduling algorithm, i.e., how to find parameters for algorithms finding desired performance-energy configurations automatically such as monitoring/tuning windows [10] but also system and application parameters [144] for thorough optimization of a run.
6. Consideration of (mixed) precision vs. energy and (mixed) precision vs. performance combined with energy trade-offs. This is especially important for ML applications deployed in a GPU environment, where single or even half float precision can be used to deliver reasonable results and the underlying hardware can provide a significant boost to the performance and/or energy efficiency.
7. The testbed environments, especially for HPC, where the hardware technologies are extremely advanced, and therefore expensive, often cannot be easily used for experiments. Thus, for existing simulators that consider performance and energy during scheduling—such as MERPSYS [145]—energy and time accuracy vs. simulation time functions shall be developed.
8. Several algorithms relatively recently deployed for scheduling optimization such as deep reinforcement learning [146] shall consider energy aspects in the future. The trend of replacing classical programming constructs with ML alternatives is spreading around all computer science areas [147], especially for problems requiring heuristics. Thus, it seems to be reasonable to assume that scheduling, and specifically its energy-aware version will be more and more supported by such an approach.
9. Since the main technology used for controlling the power lever is DVFS, we see a need for consideration of power caps in energy-aware scheduling, which apart from frequency scaling, uses other complementary techniques, e.g., thread throttling. Similarly to DVFS, it can be extensively used for heterogeneous cluster systems such as CPU+GPU systems—both single nodes and clusters, in the context of the dynamic application of power caps. This is especially relevant as more and more hardware accelerators are being proposed [2].
10. Consideration of an extended scheduling problem in which additional performance-energy configurations are considered, i.e., those that result from consideration of computing device's performance for various power caps, as shown, e.g., in [10] for CPUs and in [42,148] for GPUs.

The most promising area where there is a lack of a solution seems to be an auto configurable power capping system, based on a hybrid approach—optimized thread scheduling with power limitation by tools such as RAPL on CPU + GPU heterogeneous systems, also with consideration of resource and network contention both within a single node but also the interconnect in a cluster. Autoconfiguration may be based on auto tuning approaches [144]. The key relationship between the deviations of energy consumption measurements with the use of the software tools and the physical meter to the size of the input data also remains an open area.

Author Contributions: Conceptualization, B.K., P.C. and J.P.; methodology, P.C.; validation, B.K., P.C. and J.P.; formal analysis, B.K., P.C. and J.P.; investigation, B.K., P.C. and J.P.; writing—original draft preparation, B.K., P.C. and J.P.; writing—review and editing, B.K., P.C. and J.P.; supervision, P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following more significant abbreviations are used in this manuscript:

HPC	High-Performance Computing
ExecT	Execution Time
EC	Energy Consumption
DAG	Directed Acyclic Graph
PpW	Performance per watt
RPpW	Reference Performance per watt
TGI	The Green Index
REE	Relative Energy Efficiency
EDP	Energy Delay Product
EDS	Energy Delay Summation
EDD	Energy Delay Distance
GPU	Graphical Processing Unit
CPU	Central Processing Unit
DVFS	Dynamic Voltage and Frequency Scaling
DVS	Dynamic Voltage Scaling
DPM	Dynamic Power Management
ML	Machine Learning
RL	Reinforcement Learning
CP	Constraint Programming
GA	Genetic Algorithm
ILP	Integer Linear Programming
LP	Linear Programming
(O)	Optimized
(C)	Constrained

References

1. Czarnul, P. *Parallel Programming for Modern High Performance Computing Systems*; CRC Press: Boca Raton, FL, USA, 2018; ISBN 9781138305953.
2. Dongarra, J. *HPC: Where We Are Today and a Look into the Future*; Parallel Processing and Applied Mathematics, PPAM: Gdansk, Poland, 2022.
3. Czarnul, P.; Proficz, J.; Krzywaniak, A. Energy-Aware High-Performance Computing: Survey of State-of-the-Art Tools, Techniques, and Environments. *Sci. Program.* **2019**, *2019*, 1–19.
4. Subramaniam, B.; Feng, W.C. The Green Index: A Metric for Evaluating System-Wide Energy Efficiency in HPC Systems. In Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum, Shanghai, China, 21–25 May 2012; pp. 1007–1013. https://doi.org/10.1109/IPDPSW.2012.123.
5. Laros III, J.H.; Pedretti, K.; Kelly, S.M.; Shu, W.; Ferreira, K.; Vandyke, J.; Vaughan, C. Energy delay product. In *Energy-Efficient High Performance Computing*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 51–55.
6. Martin, A.J.; Nyström, M.; Pénzes, P.I. ET 2: A metric for time and energy efficiency of computation. In *Power Aware Computing*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 293–315.
7. Chandio, A.A.; Bilal, K.; Tziritas, N.; Yu, Z.; Jiang, Q.; Khan, S.U.; Xu, C.Z. A comparative study on resource allocation and energy efficient job scheduling strategies in large-scale parallel computing systems. *Clust. Comput.* **2014**, *17*, 1349–1367.
8. Sheikh, H.F.; Tan, H.; Ahmad, I.; Ranka, S.; Bv, P. Energy- and Performance-Aware Scheduling of Tasks on Parallel and Distributed Systems. *J. Emerg. Technol. Comput. Syst.* **2012**, *8*, 1–37. https://doi.org/10.1145/2367736.2367743.
9. Ilsche, T.; Schöne, R.; Schuchart, J.; Hackenberg, D.; Simon, M.; Georgiou, Y.; Nagel, W.E. Power measurement techniques for energy-efficient computing: reconciling scalability, resolution, and accuracy. *SICS Softw.-Intensive Cyber-Phys. Syst.* **2019**, *34*, 45–52.
10. Krzywaniak, A.; Czarnul, P.; Proficz, J. DEPO: A dynamic energy-performance optimizer tool for automatic power capping for energy efficient high-performance computing. *Softw. Pract. Exp.* **2022**, *52*, 2598–2634. https://doi.org/10.1002/spe.3139.
11. Cai, C.; Wang, L.; Khan, S.U.; Tao, J. Energy-Aware High Performance Computing: A Taxonomy Study. In Proceedings of the 2011 IEEE 17th International Conference on Parallel and Distributed Systems, Tainan, Taiwan, 7–9 December 2011; pp. 953–958. https://doi.org/10.1109/ICPADS.2011.59.
12. Benedict, S. Energy-aware performance analysis methodologies for HPC architectures—An exploratory study. *J. Netw. Comput. Appl.* **2012**, *35*, 1709–1719. https://doi.org/10.1016/j.jnca.2012.08.003.
13. Maiterth, M.; Koenig, G.; Pedretti, K.; Jana, S.; Bates, N.; Borghesi, A.; Montoya, D.; Bartolini, A.; Puzovic, M. Energy and Power Aware Job Scheduling and Resource Management: Global Survey—Initial Analysis. In Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Vancouver, BC, Canada, 21–25 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 685–693. https://doi.org/10.1109/IPDPSW.2018.00111.
14. Chaudhry, M.T.; Ling, T.C.; Manzoor, A.; Hussain, S.A.; Kim, J. Thermal-Aware Scheduling in Green Data Centers. *ACM Comput. Surv.* **2015**, *47*, 1–48. https://doi.org/10.1145/2678278.
15. Juarez, F.; Ejarque, J.; Badia, R.M. Dynamic energy-aware scheduling for parallel task-based application in cloud computing. *Future Gener. Comput. Syst.* **2018**, *78*, 257–271.

16. Rizvandi, N.B.; Taheri, J.; Zomaya, A.Y.; Lee, Y.C. Linear combinations of dvfs-enabled processor frequencies to modify the energy-aware scheduling algorithms. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, 17–20 May 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 388–397.
17. Sinnen, O. *Task Scheduling for Parallel Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
18. Kafil, M.; Ahmad, I. Optimal task assignment in heterogeneous distributed computing systems. *IEEE Concurr.* **1998**, *6*, 42–50. https://doi.org/10.1109/4434.708255.
19. Dorronsoro, B.; Pinel, F. Combining Machine Learning and Genetic Algorithms to Solve the Independent Tasks Scheduling Problem. In Proceedings of the 2017 3rd IEEE International Conference on Cybernetics (CYBCONF), Exeter, UK, 21–23 June 2017; pp. 1–8. https://doi.org/10.1109/CYBConf.2017.7985766.
20. Pietri, I.; Sakellariou, R. Energy-Aware Workflow Scheduling Using Frequency Scaling. In Proceedings of the 2014 43rd International Conference on Parallel Processing Workshops, Minneapolis, MN, USA, 9–12 Septembe 2014; pp. 104–113. https://doi.org/10.1109/ICPPW.2014.26.
21. Topcuoglu, H.; Hariri, S.; Wu, M.Y. Task scheduling algorithms for heterogeneous processors. In Proceedings of the Eighth Heterogeneous Computing Workshop (HCW'99), San Juan, PR, USA, 12 April 1999; pp. 3–14. https://doi.org/10.1109/HCW.1999.765092.
22. Bhuiyan, A.; Guo, Z.; Saifullah, A.; Guan, N.; Xiong, H. Energy-Efficient Real-Time Scheduling of DAG Tasks. *ACM Trans. Embed. Comput. Syst.* **2018**, *17*, 1–25. https://doi.org/10.1145/3241049.
23. Bambagini, M.; Marinoni, M.; Aydin, H.; Buttazzo, G. Energy-Aware Scheduling for Real-Time Systems: A Survey. *ACM Trans. Embed. Comput. Syst.* **2016**, *15*, 1–34. https://doi.org/10.1145/2808231.
24. Zeng, Q.; Du, Y.; Huang, K.; Leung, K.K. Energy-Efficient Radio Resource Allocation for Federated Edge Learning. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020; pp. 1–6. https://doi.org/10.1109/ICCWorkshops49005.2020.9145118.
25. Ravi, V.T.; Becchi, M.; Jiang, W.; Agrawal, G.; Chakradhar, S. Scheduling concurrent applications on a cluster of cpu-gpu nodes. In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), Ottawa, ON, Canada, 13–16 May 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 140–147.
26. Kim, J.K.; Siegel, H.J.; Maciejewski, A.A.; Eigenmann, R. Dynamic Resource Management in Energy Constrained Heterogeneous Computing Systems Using Voltage Scaling. *IEEE Trans. Parallel Distrib. Syst.* **2008**, *19*, 1445–1457. https://doi.org/10.1109/TPDS.2008.113.
27. Xiao, Z.; Song, W.; Chen, Q. Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 1107–1117. https://doi.org/10.1109/TPDS.2012.283.
28. Czarnul, P.; Rościszewski, P. Optimization of Execution Time under Power Consumption Constraints in a Heterogeneous Parallel System with GPUs and CPUs. In Proceedings of the 15th International Conference on Distributed Computing and Networking (ICDCN), Coimbatore, India, 4–7 January 2014.
29. Boiński, T.; Czarnul, P. Optimization of Data Assignment for Parallel Processing in a Hybrid Heterogeneous Environment Using Integer Linear Programming. *Comput. J.* **2021**, *65*, 1412–1433.
30. Kar, I.; Parida, R.R.; Das, H. Energy aware scheduling using genetic algorithm in cloud data centers. In Proceedings of the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 3–5 March 2016; pp. 3545–3550. https://doi.org/10.1109/ICEEOT.2016.7755364.
31. Koomey, J.; Berard, S.; Sanchez, M.; Wong, H. Implications of historical trends in the electrical efficiency of computing. *IEEE Ann. Hist. Comput.* **2010**, *33*, 46–54.
32. Abdulsalam, S.; Zong, Z.; Gu, Q.; Qiu, M. Using the Greenup, Powerup, and Speedup metrics to evaluate software energy efficiency. In Proceedings of the 2015 Sixth International Green and Sustainable Computing Conference (IGSC), Las Vegas, NV, USA, 14–16 December 2015; pp. 1–8. https://doi.org/10.1109/IGCC.2015.7393699.
33. Gonzalez, R.; Horowitz, M. Energy dissipation in general purpose microprocessors. *IEEE J. Solid-State Circuits* **1996**, *31*, 1277–1284. https://doi.org/10.1109/4.535411.
34. Roberts, S.I.; Wright, S.A.; Fahmy, S.A.; Jarvis, S.A. Metrics for Energy-Aware Software Optimisation. In Proceedings of the High Performance Computing: 32nd International Conference, ISC High Performance 2017, Frankfurt, Germany, 18–22 June 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 413–430. https://doi.org/10.1007/978-3-319-58667-0_22.
35. Benini, L.; Bogliolo, A.; De Micheli, G. A survey of design techniques for system-level dynamic power management. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2000**, *8*, 299–316.
36. Darwish, T.; Bayoumi, M. 5—Trends in Low-Power VLSI Design. In *The Electrical Engineering Handbook*; CHEN, W.K., Ed.; Academic Press: Burlington, MA, USA, 2005; pp. 263–280. https://doi.org/10.1016/B978-012170960-0/50022-0.
37. Safari, M.; Khorsand, R. Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment. *Simul. Model. Pract. Theory* **2018**, *87*, 311–326. https://doi.org/10.1016/j.simpat.2018.07.006.
38. Petoumenos, P.; Mukhanov, L.; Wang, Z.; Leather, H.; Nikolopoulos, D.S. Power capping: What works, what does not. In Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), Melbourne, Australia, 14–17 December 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 525–534.
39. Tsuzuku, K.; Endo, T. Power capping of CPU-GPU heterogeneous systems using power and performance models. In Proceedings of the 2015 International Conference on Smart Cities and Green ICT Systems (SMARTGREENS), Lisbon, Portugal, 20–22 May 2015; pp. 1–8.

40. Komoda, T.; Hayashi, S.; Nakada, T.; Miwa, S.; Nakamura, H. Power capping of CPU-GPU heterogeneous systems through coordinating DVFS and task mapping. In Proceedings of the 2013 IEEE 31st International Conference on Computer Design (ICCD), Asheville, NC, USA, 6–9 October 2013; pp. 349–356. https://doi.org/10.1109/ICCD.2013.6657064.
41. Borghesi, A.; Collina, F.; Lombardi, M.; Milano, M.; Benini, L. Power Capping in High Performance Computing Systems. In *Principles and Practice of Constraint Programming*; Pesant, G., Ed.; Springer International Publishing: Cham, Switzerland, 2015; pp. 524–540.
42. Krzywaniak, A.; Czarnul, P. Performance/Energy Aware Optimization of Parallel Applications on GPUs Under Power Capping. In *Parallel Processing and Applied Mathematics*; Wyrzykowski, R., Deelman, E., Dongarra, J., Karczewski, K., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 123–133.
43. Haidar, A.; Jagode, H.; Vaccaro, P.; Yarkhan, A.; Tomov, S.; Dongarra, J. Investigating power capping toward energy-efficient scientific applications. *Concurr. Comput. Pract. Exp.* **2018**, *31*, e4485. https://doi.org/10.1002/cpe.4485.
44. Imes, C.; Zhang, H.; Zhao, K.; Hoffmann, H. CoPPer: Soft Real-Time Application Performance Using Hardware Power Capping. In Proceedings of the 2019 IEEE International Conference on Autonomic Computing (ICAC), Umea, Sweden, 16–20 June 2019; pp. 31–41. https://doi.org/10.1109/ICAC.2019.00015.
45. Ramesh, S.; Perarnau, S.; Bhalachandra, S.; Malony, A.D.; Beckman, P. Understanding the Impact of Dynamic Power Capping on Application Progress. In Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rio de Janeiro, Brazil, 20–24 May 2019; pp. 793–804. https://doi.org/10.1109/IPDPS.2019.00088.
46. Berral, J.L.; Goiri, I.n.; Nou, R.; Julià, F.; Guitart, J.; Gavaldà, R.; Torres, J. Towards Energy-Aware Scheduling in Data Centers Using Machine Learning. In Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking; Association for Computing Machinery, New York, NY, USA, 13–15 April 2010; e-Energy '10; pp. 215–224. https://doi.org/10.1145/1791314.1791349.
47. Zhao, X.; Jamali, N. Energy-aware resource allocation for multicores with per-core frequency scaling. *J. Internet Serv. Appl.* **2014**, *5*, 9. https://doi.org/10.1186/s13174-014-0009-x.
48. Rajagopal, D.; Tafani, D.; Georgiou, Y.; Glesser, D.; Ott, M. A Novel Approach for Job Scheduling Optimizations Under Power Cap for ARM and Intel HPC Systems. In Proceedings of the 2017 IEEE 24th International Conference on High Performance Computing (HiPC), Jaipur, India, 18–21 December 2017; pp. 142–151. https://doi.org/10.1109/HiPC.2017.00025.
49. Borghesi, A.; Bartolini, A.; Lombardi, M.; Milano, M.; Benini, L. Scheduling-based power capping in high performance computing systems. *Sustain. Comput. Inform. Syst.* **2018**, *19*, 1–13. https://doi.org/10.1016/j.suscom.2018.05.007.
50. Zhang, Z.; Lang, M.; Pakin, S.; Fu, S. Trapped capacity: Scheduling under a power cap to maximize machine-room throughput. In Proceedings of the 2014 Energy Efficient Supercomputing Workshop, New Orleans, LA, USA, 16–21 November 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 41–50.
51. Nair, P.P.; Devaraj, R.; Sarkar, A. FEST: Fault-Tolerant Energy-Aware Scheduling on Two-Core Heterogeneous Platform. In Proceedings of the 2018 8th International Symposium on Embedded Computing and System Design (ISED), Cochin, India, 13–15 December 2018; pp. 63–68. https://doi.org/10.1109/ISED.2018.8704123.
52. Goiri, I.; Julià, F.; Nou, R.; Berral, J.L.; Guitart, J.; Torres, J. Energy-Aware Scheduling in Virtualized Datacenters. In Proceedings of the 2010 IEEE International Conference on Cluster Computing, Heraklion, Greece, 20–24 September 2010; pp. 58–67. https://doi.org/10.1109/CLUSTER.2010.15.
53. Zhu, X.; Yang, L.; Chen, H.; Wang, J.; Yin, S.; Liu, X. Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds. *Cloud Comput. IEEE Trans.* **2014**, *2*, 168–180. https://doi.org/10.1109/TCC.2014.2310452.
54. Hosseinimotlagh, S.; Khunjush, F.; Hosseinimotlagh, S. A Cooperative Two-Tier Energy-Aware Scheduling for Real-Time Tasks in Computing Clouds. In Proceedings of the 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Turin, Italy, 12–14 February 2014; pp. 178–182. https://doi.org/10.1109/PDP.2014.91.
55. Ardagna, D.; Panicucci, B.; Trubian, M.; Zhang, L. Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments. *IEEE Trans. Serv. Comput.* **2012**, *5*, 2–19. https://doi.org/10.1109/TSC.2010.42.
56. Kandhalu, A.; Kim, J.; Lakshmanan, K.; Rajkumar, R. Energy-Aware Partitioned Fixed-Priority Scheduling for Chip Multiprocessors. In Proceedings of the 2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications, Toyama, Japan, 28–31 August 2011; Volume 1, pp. 93–102. https://doi.org/10.1109/RTCSA.2011.75.
57. D'Amico, M.; Gonzalez, J.C. Energy hardware and workload aware job scheduling towards interconnected HPC environments. *IEEE Trans. Parallel Distrib. Syst.* **2021**, 1. https://doi.org/10.1109/tpds.2021.3090334.
58. Li, D.; Byna, S.; Chakradhar, S. Energy-Aware Workload Consolidation on GPU. In Proceedings of the 2011 40th International Conference on Parallel Processing Workshops, Taipei, Taiwan, 13–16 September 2011; pp. 389–398. https://doi.org/10.1109/ICPPW.2011.25.
59. Guerreiro, J.; Ilic, A.; Roma, N.; Tomás, P. Multi-kernel Auto-Tuning on GPUs: Performance and Energy-Aware Optimization. In Proceedings of the 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Turku, Finland, 4–6 March 2015; pp. 438–445. https://doi.org/10.1109/PDP.2015.44.
60. Yao, C.; Liu, W.; Tang, W.; Hu, S. EAIS: Energy-aware adaptive scheduling for CNN inference on high-performance GPUs. *Future Gener. Comput. Syst.* **2022**, *130*, 253–268. https://doi.org/10.1016/j.future.2022.01.004.
61. Pirahandeh, M.; Kim, D.H. Energy-Aware GPU-RAID Scheduling for Reducing Energy Consumption in Cloud Storage Systems. In *Computer Science and Its Applications*; Park, J.J.J.H., Stojmenovic, I., Jeong, H.Y., Yi, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; pp. 705–711.

62. Pirahandeh, M.; Kim, D.H. EGE: A New Energy-Aware GPU Based Erasure Coding Scheduler for Cloud Storage Systems. In Proceedings of the 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), Prague, Czech Republic, 3–6 July 2018; pp. 619–621. https://doi.org/10.1109/ICUFN.2018.8436594.
63. Sun, Y.; Gong, X.; Ziabari, A.K.; Yu, L.; Li, X.; Mukherjee, S.; McCardwell, C.; Villegas, A.; Kaeli, D. Hetero-mark, a benchmark suite for CPU-GPU collaborative computing. In Proceedings of the 2016 IEEE International Symposium on Workload Characterization (IISWC), Providence, RI, USA, 25–27 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–10.
64. Rościszewski, P.; Czarnul, P.; Lewandowski, R.; Schally-Kacprzak, M. KernelHive: A New Workflow-Based Framework for Multilevel High Performance Computing Using Clusters and Workstations with CPUs and GPUs. *Concurr. Comput. Pract. Exp.* **2016**, *28*, 2586–2607. Available online: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3719 (accessed on 15 December 2022). https://doi.org/10.1002/cpe.3719.
65. Gregg, C.; Boyer, M.; Hazelwood, K.; Skadron, K. Dynamic heterogeneous scheduling decisions using historical runtime data. In Proceedings of the Workshop on Applications for Multi-and Many-Core Processors (A4MMC), San Jose, CA, USA, June 2011; pp. 1–12.
66. Czarnul, P. Investigation of Parallel Data Processing Using Hybrid High Performance CPU + GPU Systems and CUDA Streams. *Comput. Inform.* **2020**, *39*, 510–536. https://doi.org/10.31577/cai_2020_3_510.
67. Arafa, Y.; ElWazir, A.; ElKanishy, A.; Aly, Y.; Elsayed, A.; Badawy, A.H.; Chennupati, G.; Eidenbenz, S.; Santhi, N. Verified Instruction-Level Energy Consumption Measurement for NVIDIA GPUs. In Proceedings of the 17th ACM International Conference on Computing Frontiers, Bertinoro, Italy, 17–22 May 2020; Association for Computing Machinery: New York, NY, USA, 2020; CF '20; pp. 60–70. https://doi.org/10.1145/3387902.3392613.
68. Fahad, M.; Shahid, A.; Manumachu, R.R.; Lastovetsky, A. A Comparative Study of Methods for Measurement of Energy of Computing. *Energies* **2019**, *12*, 2204. https://doi.org/10.3390/en12112204.
69. Tang, X.; Fu, Z. CPU–GPU Utilization Aware Energy-Efficient Scheduling Algorithm on Heterogeneous Computing Systems. *IEEE Access* **2020**, *8*, 58948–58958. https://doi.org/10.1109/ACCESS.2020.2982956.
70. Mejri, N.; Dupont, B.; Da Costa, G. Energy-aware scheduling of malleable HPC applications using a Particle Swarm optimised greedy algorithm. *Sustain. Comput. Inform. Syst.* **2020**, *28*, 100447.
71. Angel, E.; Bampis, E.; Kacem, F. Energy Aware Scheduling for Unrelated Parallel Machines. In Proceedings of the 2012 IEEE International Conference on Green Computing and Communications, Besancon, France, 20–23 November 2012; pp. 533–540. https://doi.org/10.1109/GreenCom.2012.78.
72. Lin, X.; Wang, Y.; Pedram, M. A Reinforcement Learning-Based Power Management Framework for Green Computing Data Centers. In Proceedings of the 2016 IEEE International Conference on Cloud Engineering (IC2E), Berlin, Germany, 4–8 April 2016; pp. 135–138. https://doi.org/10.1109/IC2E.2016.33.
73. Li, J.; Zhang, X.; Wei, Z.; Wei, J.; Ji, Z. Energy-aware task scheduling optimization with deep reinforcement learning for large-scale heterogeneous systems. *CCF Trans. High Perform. Comput.* **2021**, *3*, 383–392.
74. Imes, C.; Hofmeyr, S.; Hoffmann, H. Energy-efficient application resource scheduling using machine learning classifiers. In Proceedings of the 47th International Conference on Parallel Processing, Eugene, OR, USA, 13–16 August 2018; pp. 1–11.
75. Bellman, R. The theory of dynamic programming. *Bull. Am. Math. Soc.* **1954**, *60*, 503–515.
76. Bellman, R.E.; Dreyfus, S.E. *Applied Dynamic Programming*; Princetown University Press: Princeton, NJ, USA, 1962.
77. Li, Y.; Niu, J.; Atiquzzaman, M.; Long, X. Energy-aware scheduling on heterogeneous multi-core systems with guaranteed probability. Special Issue on Scalable Cyber-Physical Systems. *J. Parallel Distrib. Comput.* **2017**, *103*, 64–76. https://doi.org/10.1016/j.jpdc.2016.11.014.
78. Chen, J.; John, L.K. Energy-aware application scheduling on a heterogeneous multi-core system. In Proceedings of the 2008 IEEE International Symposium on Workload Characterization, Seattle, WA, USA, 14–16 September 2008; pp. 5–13. https://doi.org/10.1109/IISWC.2008.4636086.
79. Zadeh, L. Fuzzy logic. *Computer* **1988**, *21*, 83–93. https://doi.org/10.1109/2.53.
80. Méndez-Díaz, I.; Orozco, J.; Santos, R.; Zabala, P. Energy-aware scheduling mandatory/optional tasks in multicore real-time systems. *Int. Trans. Oper. Res.* **2017**, *24*, 173–198.
81. Keller, J.; Litzinger, S. Systematic search space design for energy-efficient static scheduling of moldable tasks. *J. Parallel Distrib. Comput.* **2022**, *162*, 44–58.
82. Etinski, M.; Corbalan, J.; Labarta, J.; Valero, M. Parallel job scheduling for power constrained HPC systems. *Parallel Comput.* **2012**, *38*, 615–630.
83. Tang, Q.; Gupta, S.K.S.; Varsamopoulos, G. Energy-Efficient Thermal-Aware Task Scheduling for Homogeneous High-Performance Computing Data Centers: A Cyber-Physical Approach. *IEEE Trans. Parallel Distrib. Syst.* **2008**, *19*, 1458–1472. https://doi.org/10.1109/TPDS.2008.111.
84. Agrawal, P.; Rao, S. Energy-aware scheduling of distributed systems. *IEEE Trans. Autom. Sci. Eng.* **2014**, *11*, 1163–1175.
85. Mezmaz, M.; Melab, N.; Kessaci, Y.; Lee, Y.C.; Talbi, E.G.; Zomaya, A.Y.; Tuyttens, D. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *J. Parallel Distrib. Comput.* **2011**, *71*, 1497–1508.
86. Lee, Y.C.; Zomaya, A.Y. Minimizing Energy Consumption for Precedence-Constrained Applications Using Dynamic Voltage Scaling. In Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai, China, 18–21 May 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 92–99. https://doi.org/10.1109/CCGRID.2009.16.

87. Guzek, M.; Pecero, J.E.; Dorronsoro, B.; Bouvry, P. Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems. *Appl. Soft Comput.* **2014**, *24*, 432–446.
88. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. https://doi.org/10.1109/4235.996017.
89. Nebro, A.J.; Durillo, J.J.; Luna, F.; Dorronsoro, B.; Alba, E. Design Issues in a Multiobjective Cellular Genetic Algorithm. In *Evolutionary Multi-Criterion Optimization*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4403 LNCS, pp. 126–140. https://doi.org/10.1007/978-3-540-70928-2_13.
90. Zitzler, E.; Künzli, S. Indicator-Based Selection in Multiobjective Search. In *Parallel Problem Solving from Nature—PPSN VIII*; Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.P., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 832–842.
91. Kolodziej, J.; Khan, S.U.; Xhafa, F. Genetic algorithms for energy-aware scheduling in computational grids. In Proceedings of the 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Barcelona, Spain, 8–10 November 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 17–24.
92. Kassab, A.; Nicod, J.M.; Philippe, L.; Rehn-Sonigo, V. Assessing the use of genetic algorithms to schedule independent tasks under power constraints. In Proceedings of the 2018 International Conference on High Performance Computing & Simulation (HPCS), Orleans, France, 16–20 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 252–259.
93. Kassab, A.; Nicod, J.m.; Philippe, L.; Rehn-Sonigo, V. Scheduling Independent Tasks in Parallel under Power Constraints. In Proceedings of the 2017 46th International Conference on Parallel Processing (ICPP), Bristol, UK, 14–17 August 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 543–552. https://doi.org/10.1109/ICPP.2017.63.
94. Mishra, R.; Rastogi, N.; Zhu, D.; Mossé, D.; Melhem, R. Energy aware scheduling for distributed real-time systems. In Proceedings of the International Parallel and Distributed Processing Symposium, Cambridge, MA, USA, 20–24 May 2003; IEEE: Piscataway, NJ, USA, 2003; p. 9.
95. Chiesi, M.; Vanzolini, L.; Mucci, C.; Franchi Scarselli, E.; Guerrieri, R. Power-Aware Job Scheduling on Heterogeneous Multicore Architectures. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 868–877. https://doi.org/10.1109/TPDS.2014.2315203.
96. Wang, L.; Von Laszewski, G.; Dayal, J.; Wang, F. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, 17–20 May 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 368–377.
97. Nesmachnow, S.; Dorronsoro, B.; Pecero, J.E.; Bouvry, P. Energy-aware scheduling on multicore heterogeneous grid computing systems. *J. Grid Comput.* **2013**, *11*, 653–680.
98. Aupy, G.; Benoit, A.; Robert, Y. Energy-aware scheduling under reliability and makespan constraints. In Proceedings of the 2012 19th International Conference on High Performance Computing, Pune, India, 18–22 December 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1–10.
99. Yang, X.; Zhou, Z.; Wallace, S.; Lan, Z.; Tang, W.; Coghlan, S.; Papka, M.E. Integrating dynamic pricing of electricity into energy aware scheduling for HPC systems. In Proceedings of the SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 17–21 November 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–11.
100. Barik, R.; Farooqui, N.; Lewis, B.T.; Hu, C.; Shpeisman, T. A Black-Box Approach to Energy-Aware Scheduling on Integrated CPU-GPU Systems. In Proceedings of the 2016 International Symposium on Code Generation and Optimization, Barcelona, Spain, 12–18 March 2016; Association for Computing Machinery: New York, NY, USA, 2016; CGO '16; pp. 70–81. https://doi.org/10.1145/2854038.2854052.
101. Maheswaran, M.; Ali, S.; Siegel, H.J.; Hensgen, D.; Freund, R.F. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *J. Parallel Distrib. Comput.* **1999**, *59*, 107–131.
102. Li, Y.; Liu, Y.; Qian, D. A Heuristic Energy-aware Scheduling Algorithm for Heterogeneous Clusters. In Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems, Shenzhen, China, 9–11 December 2009; pp. 407–413. https://doi.org/10.1109/ICPADS.2009.33.
103. Biswas, T.; Kuila, P.; Ray, A.K. A novel energy efficient scheduling for high performance computing systems. In Proceedings of the 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 10–12 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
104. Amalarethinam, D.G.; Kavitha, S. Priority based performance improved algorithm for meta-task scheduling in cloud environment. In Proceedings of the 2017 2nd International Conference on Computing and Communications Technologies (ICCCT), Chennai, India, 23–24 February 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 69–73.
105. Mei, J.; Li, K. Energy-Aware Scheduling Algorithm with Duplication on Heterogeneous Computing Systems. In Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing, Beijing, China, 20–23 September 2012; pp. 122–129. https://doi.org/10.1109/Grid.2012.32.
106. Bansal, S.; Kumar, P.; Singh, K. Dealing with heterogeneity through limited duplication for scheduling precedence constrained task graphs. *J. Parallel Distrib. Comput.* **2005**, *65*, 479–491.
107. Hagras, T.; Janecek, J. A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems. In Proceedings of the 18th International Parallel and Distributed Processing Symposium, Santa Fe, NM, USA, 26–30 April 2004; p. 107.

108. Zong, Z.; Manzanares, A.; Ruan, X.; Qin, X. EAD and PEBD: Two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters. *IEEE Trans. Comput.* **2010**, *60*, 360–374.
109. Ebaid, A.; Rajasekaran, S.; Ammar, R.; Ebaid, R. Energy-aware heuristics for scheduling parallel applications on high performance computing platforms. In Proceedings of the 2014 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Noida, India, 15–17 December 2014; pp. 282–289. https://doi.org/10.1109/ISSPIT.2014.7300601.
110. Ranaweera, S.; Agrawal, D.P. A task duplication based scheduling algorithm for heterogeneous systems. In Proceedings of the 14th International Parallel and Distributed Processing Symposium, IPDPS 2000, Cancun, Mexico, 1–5 May 2000; IEEE: Piscataway, NJ, USA, 2000; pp. 445–450.
111. Mämmelä, O.; Majanen, M.; Basmadjian, R.; Meer, H.; Giesler, A.; Homberg, W. Energy-aware job scheduler for high-performance computing. *Comput. Sci.-Res. Dev.* **2012**, *27*, 265–275. https://doi.org/10.1007/s00450-011-0189-6.
112. Mashayekhy, L.; Nejad, M.M.; Grosu, D.; Lu, D.; Shi, W. Energy-aware scheduling of mapreduce jobs. In Proceedings of the 2014 IEEE International Congress on Big Data, Washington, DC, USA, 27–30 October 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 32–39.
113. Mashayekhy, L.; Nejad, M.M.; Grosu, D.; Zhang, Q.; Shi, W. Energy-Aware Scheduling of MapReduce Jobs for Big Data Applications. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 2720–2733. https://doi.org/10.1109/TPDS.2014.2358556.
114. Mei, X.; Wang, Q.; Chu, X.; Liu, H.; Leung, Y.W.; Li, Z. Energy-aware Task Scheduling with Deadline Constraint in DVFS-enabled Heterogeneous Clusters. *arXiv* **2021**, arXiv:2104.00486. https://doi.org/10.48550/ARXIV.2104.00486.
115. Kiselev, E.; Telegin, P.N.; Shabanov, B.M. An energy-efficient scheduling algorithm for shared facility supercomputer centers. *Lobachevskii J. Math.* **2021**, *42*, 2554–2561.
116. Wong, P.; Der Wijngaart, R. NAS parallel benchmarks I/O version 2.4. In *Technical Report NAS-03-002*; NASA Ames Research Center: Moffet Field, CA, USA, 2003; p. 91.
117. Hu, Y.; Liu, C.; Li, K.; Chen, X.; Li, K. Slack allocation algorithm for energy minimization in cluster systems. *Future Gener. Comput. Syst.* **2017**, *74*, 119–131.
118. Maurya, A.K.; Modi, K.; Kumar, V.; Naik, N.S.; Tripathi, A.K. Energy-aware scheduling using slack reclamation for cluster systems. *Clust. Comput.* **2020**, *23*, 911–923.
119. Baskiyar, S.; Abdel-Kader, R. Energy aware DAG scheduling on heterogeneous systems. *Clust. Comput.* **2010**, *13*, 373–383.
120. Park, G.L.; Shirazi, B.; Marquis, J.; Choo, H. Decisive path scheduling: A new list scheduling method. In Proceedings of the Proceedings of the 1997 International Conference on Parallel Processing (Cat. No. 97TB100162), Bloomington, IL, USA, 11–15 August 1997; IEEE: Piscataway, NJ, USA, 1997, pp. 472–480.
121. Roeder, J.; Rouxel, B.; Altmeyer, S.; Grelck, C. Energy-aware scheduling of multi-version tasks on heterogeneous real-time systems. In Proceedings of the 36th Annual ACM Symposium on Applied Computing, Gwangju, Republic of Korea, 22–26 March 2021; pp. 501–510.
122. Ebrahimirad, V.; Goudarzi, M.; Rajabi, A. Energy-aware scheduling for precedence-constrained parallel virtual machines in virtualized data centers. *J. Grid Comput.* **2015**, *13*, 233–253.
123. Topcuoglu, H.; Hariri, S.; Wu, M.Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* **2002**, *13*, 260–274.
124. Li, K. Energy efficient scheduling of parallel tasks on multiprocessor computers. *J. Supercomput.* **2012**, *60*, 223–247.
125. Chen, J.; He, Y.; Zhang, Y.; Han, P.; Du, C. Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems. *J. Syst. Archit.* **2022**, *129*, 102598.
126. Shekar, V.; Izadi, B. Energy aware scheduling for DAG structured applications on heterogeneous and DVS enabled processors. In Proceedings of the International Conference on Green Computing, Chicago, IL, USA, 15–18 August 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 495–502.
127. Sih, G.C.; Lee, E.A. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. *IEEE Trans. Parallel Distrib. Syst.* **1993**, *4*, 175–187.
128. Sun, H.; Stolf, P.; Pierson, J.M. Spatio-temporal thermal-aware scheduling for homogeneous high-performance computing datacenters. *Future Gener. Comput. Syst.* **2017**, *71*, 157–170.
129. Raghu, H.; Saurav, S.K.; Bapu, B.S. PAAS: Power Aware Algorithm for Scheduling in High Performance Computing. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, Washington, DC, USA, 9–12 December 2013; pp. 327–332. https://doi.org/10.1109/UCC.2013.71.
130. Hu, Y.; Li, J.; He, L. A reformed task scheduling algorithm for heterogeneous distributed systems with energy consumption constraints. *Neural Comput. Appl.* **2020**, *32*, 5681–5693.
131. Lee, Y.C.; Zomaya, A.Y. Energy Conscious Scheduling for Distributed Computing Systems under Different Operating Conditions. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 1374–1381. https://doi.org/10.1109/TPDS.2010.208.
132. Bozdag, D.; Catalyurek, U.; Ozguner, F. A task duplication based bottom-up scheduling algorithm for heterogeneous environments. In Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium, Rhodes, Greece, 25–29 April 2006; IEEE: Piscataway, NJ, USA, 2006; p. 12.
133. MA, Y.; GONG, B.; GUO, Z.; CHEN, Y.; ZOU, L. Energy-aware scheduling of parallel application in hybrid computing system. *Chin. J. Electron.* **2014**, *23*, 688–694.
134. Chen, S.; Li, Z.; Yang, B.; Rudolph, G. Quantum-Inspired Hyper-Heuristics for Energy-Aware Scheduling on Heterogeneous Computing Systems. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 1796–1810. https://doi.org/10.1109/TPDS.2015.2462835.

135. Deng, Z.; Yan, Z.; Huang, H.; Shen, H. Energy-Aware Task Scheduling on Heterogeneous Computing Systems With Time Constraint. *IEEE Access* **2020**, *8*, 23936–23950. https://doi.org/10.1109/ACCESS.2020.2970166.
136. Li, K.; Tang, X.; Li, K. Energy-Efficient Stochastic Task Scheduling on Heterogeneous Computing Systems. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 2867–2876. https://doi.org/10.1109/TPDS.2013.270.
137. Kim, K.H.; Buyya, R.; Kim, J. Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters. In Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07), Rio de Janeiro, Brazil, 14–17 May 2007; pp. 541–548. https://doi.org/10.1109/CCGRID.2007.85.
138. Cong, J.; Yuan, B. Energy-Efficient Scheduling on Heterogeneous Multi-Core Architectures. In Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design, Redondo Beach, CA, USA, 30 July 2012–1 August 2012; Association for Computing Machinery: New York, NY, USA, 2012; ISLPED '12; pp. 345–350. https://doi.org/10.1145/2333660.2333737.
139. Sawalha, L.; Barnes, R.D. Energy-Efficient Phase-Aware Scheduling for Heterogeneous Multicore Processors. In Proceedings of the 2012 IEEE Green Technologies Conference, Besancon, France, 20–23 November 2012; pp. 1–6. https://doi.org/10.1109/GREEN.2012.6200965.
140. Manumachu, R.R.; Lastovetsky, A. Bi-Objective Optimization of Data-Parallel Applications on Homogeneous Multicore Clusters for Performance and Energy. *IEEE Trans. Comput.* **2018**, *67*, 160–177. https://doi.org/10.1109/TC.2017.2742513.
141. Khaleghzadeh, H.; Fahad, M.; Shahid, A.; Manumachu, R.R.; Lastovetsky, A. Bi-Objective Optimization of Data-Parallel Applications on Heterogeneous HPC Platforms for Performance and Energy through Workload Distribution. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 543–560. https://doi.org/10.1109/TPDS.2020.3027338.
142. Khaleghzadeh, H.; Reddy Manumachu, R.; Lastovetsky, A. Efficient Exact Algorithms for Continuous Bi-Objective Performance-Energy Optimization of Applications with Linear Energy and Monotonically Increasing Performance Profiles on Heterogeneous High Performance Computing Platforms. *Concurr. Comput. Pract. Exp.* **2022**, e7285. Available online: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.7285 (accessed on 15 December 2022). https://doi.org/10.1002/cpe.7285.
143. Li, D.; Wu, J. Energy-aware scheduling for frame-based tasks on heterogeneous multiprocessor platforms. In Proceedings of the 2012 41st International Conference on Parallel Processing, Pittsburgh, PA, USA, 10–13 September 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 430–439.
144. Czarnul, P.; Rościszewski, P. Auto-tuning methodology for configuration and application parameters of hybrid CPU + GPU parallel systems based on expert knowledge. In Proceedings of the 2019 International Conference on High Performance Computing Simulation (HPCS), Dublin, Ireland, 15–19 July 2019; pp. 551–558. https://doi.org/10.1109/HPCS48598.2019.9188060.
145. Czarnul, P.; Kuchta, J.; Matuszek, M.R.; Proficz, J.; Rosciszewski, P.; Wójcik, M.; Szymanski, J. MERPSYS: An environment for simulation of parallel application execution on large scale HPC systems. *Simul. Model. Pract. Theory* **2017**, *77*, 124–140. https://doi.org/10.1016/j.simpat.2017.05.009.
146. Jaime Fomperosa, Mario Ibañez, E.S.; Bosque, J.L. Task Scheduler for Heterogeneous Data Centres based on Deep Reinforcement Learning. In *Parallel Processing and Applied Mathematics*; PPAM: Gdansk, Poland, 2022.
147. Welsh, M. The End of Programming. *Commun. ACM* **2023**, *66*, 34–35. https://doi.org/10.1145/3570220.
148. Krzywaniak, A.; Czarnul, P.; Proficz, J. GPU Power Capping for Energy-Performance Trade-Offs in Training of Deep Convolutional Neural Networks for Image Recognition. In Proceedings of the Computational Science—ICCS 2022: 22nd International Conference, Part I, London, UK, 21–23 June 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 667–681. https://doi.org/10.1007/978-3-031-08751-6_48.