Towards Energy-aware Scheduling of Scientific Workflows

Mehul Warade
School of Information Technology
Deakin University
Geelong, Victoria, 3220, Australia
mehul.warade@research.deakin.edu.au

Jean-Guy Schneider
School of Information Technology
Deakin University
Geelong, Victoria, 3220, Australia
jeanguy.schneider@deakin.edu.au

Kevin Lee
School of Information Technology
Deakin University
Geelong, Victoria, 3220, Australia
kevin.lee@deakin.edu.au

***Abstract*—Contemporary scientific computation is increasingly structured using scientific workflows that are executed on highly scalable compute clusters. The execution of these workflows is generally geared towards optimizing run-time performance with the energy footprint of the execution being ignored. However, there is evidence that minimizing both execution time as well as energy consumption are not mutually exclusive. The aim of the work presented in this paper is to highlight the benefits of energy-aware scientific workflow execution. In this paper, we propose a set of requirements for an energy-aware scheduling and present an architecture for the implementation of an energy-aware scheduler.**

***Index Terms*—Energy-Aware Computing, Scientific workflows, Scheduling**

I. Introduction

A scientific workflow is defined as series of small tasks being executed in a specific structure to achieve a certain computation goal [1]. A trade off between run-time performance and the energy consumption has been seen for multiple workflows [2]. This often means that using more cluster resources results in minimal computational improvements but large energy overheads. Workflow engines are used to execute the workflow and handle all the data, task dependencies, logging and reporting [3]–[5], respectively.

Workflow engines have been developed that provide the user with a lot of optimization and configuration options. Most scientists do not manage their own cluster infrastructure and rely on the workflow engine to handles the creation, management and debugging of the cluster. This makes it difficult for them to be able to understand the environmental impact of their computation and to optimize their workflows or the cluster for better performance and/or energy consumption.

Scientists have optimized the performance of different workflows for timeliness [6], performance [7], or data provenance [8]. These optimizations are particular to the specific workflows and are achieved by modelling and analysing the workflow [9], [10]. In existing workflow schedulers, energy considerations are only marginally (if at all) taken into account. For example, current approaches focus on executing individual jobs rather than analysing and understanding the structure and execution characteristics of the workflow and the individual jobs in the workflow [9], [11], [12].

The aim of this paper is to motivate the development of a scheduler taking into consideration the energy consumption of a scientific workflow and its underlying jobs. The scheduler needs to be able to understand the workflow at job level or computation level in order to make changes to the workflow or the cluster to better accommodate the execution of the workflow. A set of generic and workflow-specific requirements and policies for the scheduler are also identified. The authors anticipate that a scheduler built to address these requirements will reduce the energy consumption of workflows.

The key contributions of this paper are as follows: (i) a survey of the current state of energy aware workflow schedulers; (ii) identification of requirements and challenges for an energy aware schedulers; and (iii) a conceptual architecture for an energy-aware scheduler for scientific workflows.

The remainder of this paper is as follows: Section II provides an overview of the issues associated with workflow execution and energy constraints. This section also provides an analysis of popular scientific workflows. The requirements and challenges for the development of an energy-aware scheduler are documented in Section III. Section IV presents a conceptual architecture for developing an energy-aware scheduler. Finally, Section V provides conclusions and future work.

II. Background

High Performance Computing (HPC) has led to an increase in the global energy usage [13]. As we move towards ExaFLOP performance in HPC, it is becoming more and more challenging to cope with the increasing energy required for the computation [2]. Monitoring the energy usage and developing more energy efficient methods for computing is of ever increasing importance [13], [14]. A number of hardware and software methods have been proposed to tackle this issue. The remainder of this section provides a background on (i) scientific workflows, (ii) workflow schedulers, and iii) energy-aware scheduling, respectively.

A. Scientific Workflows

Workflows are increasingly being used in High Performance Computing (HPC) to accommodate for the growing complexity of computation, simulations and analysis [15]. A workflow comprises of a number of individual jobs that are executed

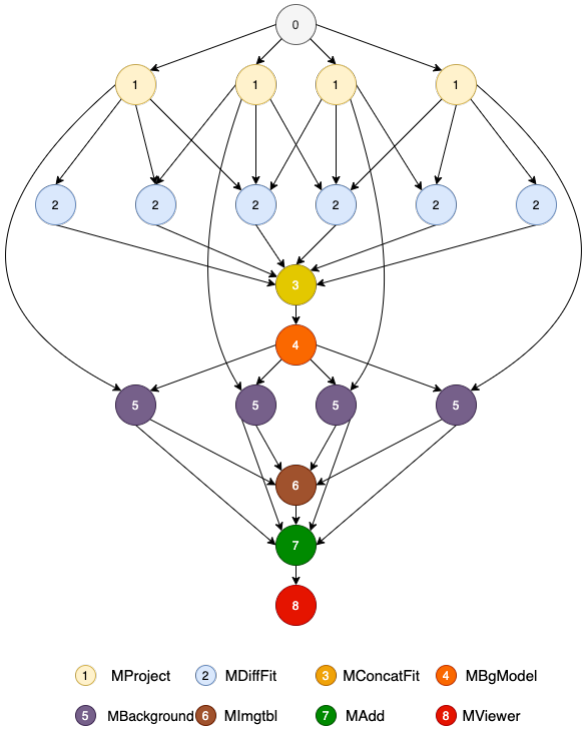

Fig. 1. A Simple Montage Workflow.

in batches to complete a large computation. These jobs have characteristics that are particular to individual workflow such as data dependencies, bottlenecks, etc. The execution of such jobs can be exploited to achieve improvement in performance or energy consumption of the workflow. In this section, two popular scientific workflows are described and the jobs that can be exploited to achieve improvements are discussed.

1) Montage: Montage [16], [17] is a software toolkit used in Astrophotography to combine Flexible Image Transport System format (FITS) images of the sky into composite images called *mosaics*. The toolkit preserves the calibration and positional fidelity of the original input images. A Montage workflow comprises of a number of tasks to develop a relevant mosaic of the sky based on its input parameters. Montage has been classified as an input/output-bound workflow [18] compared to other scientific workflows. The Montage toolkit can be used to generate workflows of varying size depending on the requirements of a scientist. The varying size of a Montage workflow is specified in (i) *degrees* of the sky and (ii) the colour channels which the final images should be generated from. Figure 1 illustrates the Montage workflow as a directed acyclic graph (DAG). The Montage DAG has 8 levels of jobs which are dependent on each other after the workflow is submitted (Step 0).

2) Bioinformatics: The Bioinformatics workflow is based on the data collected by the 1,000 Genomes Project [19], [20]. The purpose of this workflow is to analyze the data and cross-match the whole datasets for mutations. The workflow also identifies mutational overlaps in order to evaluate potential disease-related mutations. The extracted data, along with the

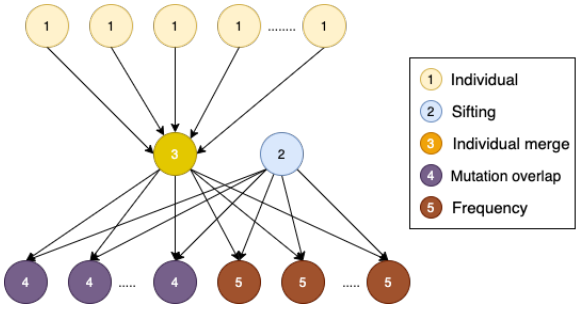

Fig. 2. A Simple Bioinformatics Workflow.

mutation's sift scores (calculated by the Variant Effect Predictor [21]) can help researchers in discovering the exact mutation which is the cause for a certain disease in a person. The Bioinformatics workflow has many characteristics which are specific to the workflow. In particular, there are two different input variables that can be controlled by the user – the size of workflow (the data to be computed) and the number of parallel jobs. Figure 2 illustrates a directed acyclic graph (DAG) of a Bioinformatics workflow.

Like any scientific workflow, both Montage and the Bioinformatics workflow have specific characteristics to their structure. Montage has some jobs, in particular `mProject`, that are computationally more intensive than others and the bioinformatics workflow have a bottleneck in its execution (`Individual_merge`). The size and parameters of the workflow dictate the number of jobs that are needed for execution. The combination of the size of workflow and the number of cluster nodes have a direct impact on the queuing of jobs, execution time, and the energy consumption of the workflow. Identifying such jobs and smartly executing them can lead to improvements in energy consumption and performance of the workflows.

B. Workflow Schedulers

Optimizing schedulers for workflows have always been an area of interest for scientists. An adaptive workflow processing and execution method is possible [22]–[24]. These methods take into account the current progress of the workflow and the load on the cluster to perform scheduling and load balancing.

Another approach for developing effective scheduling of workflows was introduced in a cloud based distributed computing systems [25]. Static Provisioning-Static Scheduling under Energy and Budget Constraints (SPSS-EB) and Static Provisioning-Static Scheduling under Energy and Deadline Constraints (SPSS-ED) were two novel algorithms developed and tested to show improvements in performance of workflows in a cloud based distributed systems [26].

A system for dynamically allocating tasks based on the available infrastructure and knowledge of the workflow have been developed [27]. The system was able to achieve faster job run-time along with improved cluster usage. Another scheduler makes use of critical path analysis to find the optimal execution of tasks to reduce the data transfer between the

nodes [12]. This scheduler achieved reduction of 66% in execution time over traditional schedulers. A hybrid algorithm making use of Particle swarm optimization (PSO) algorithm and processing bottleneck tasks on high priority have been developed to achieve better execution time along with no loss in cluster load [28]. Similar results have been obtained from a scheduler developed by using genetic algorithm [29].

C. Energy Aware Workflow Schedulers

This section focuses on schedulers that have been developed by considering the cost and energy consumption of the computation. In recent years, the focus has shifted from 'faster computation' to 'energy-efficient computation'. Due to this a lot of researchers have developed systems that take the cost of computation in consideration while scheduling the workflows.

Scheduling algorithms have also been introduced to meet time deadlines of a computation while minimizing the energy consumption [9], [30]. A scheduler based on polynomial time algorithm is proposed to provide real-time dynamic resource allocation in order to get good solutions in a particular time [31]. Chebyshev scalarization function is used to develop an energy-aware multi-objective reinforcement learning (EnMORL) algorithm to reduce the makespan and energy consumption of a workflow [32].

Cloud computing is one of the most researched area for HPC and energy savings. Not everyone can afford huge data centers and, cloud computing is go-to for any researcher looking for computation. Inter-dependency of tasks leads to huge data transfer and less computation tasks being executed in the clouds. A scheduler to reduce the data transfer and inter-dependency has been developed with the focus on reducing the energy footprint of the workflow [33]. The scheduler was able to achieve 22.7% reduction in energy at no cost to the make span of the workflow.

An Energy-Efficient Task Offloading (EETO) policy has been developed to schedule and offload real-time IoT applications [11]. The policy makes use of Lyapunov optimization technique to minimize the queuing of tasks and achieves energy consumption reduction of about 23.79% as compared to the current techniques. Similarly, a dynamic offloading and resource scheduling policy is developed to reduce energy consumption and shorten application completion time [34]. This policy dynamically optimizes the CPU clock frequency and the wireless transmission power to achieve reduction in the energy-efficiency cost (EEC) of the workflow.

The current research in the domain of energy aware scheduling are workflow specific and cannot be used for other computations in the scientific community. These scheduler do not take into account the optimizations that can be made on workflow, job and cluster level. The schedulers try to reduce the inter-dependencies, queuing, increase cluster usage, etc. A workflow can be executed in a wide number of ways on large configurations of clusters. A need for a generic scheduler arises that can exploit all the different policies to help reduce the energy footprint of the workflow. This scheduler should take into account all the combinations of cluster configuration and workflow executions to make the optimal scheduling decision.

III. Requirements and Challenges for an Energy-Aware Scheduler

In this section, the requirements for an energy-aware scheduler are presented, followed by a discussion what challenges must be overcome to implement these requirements.

A. Energy-Aware Scheduler Requirements

The primary aim of the scheduler is to reduce the energy consumption of workflow executions. To achieve this, the following requirements are identified that need to be met:

- R1 The scheduler shall attempt to reduce the energy used for executing a job, workflow or set of workflows, respectively;
- R2 The proposed scheduler shall not vary the output of the workflow;
- R3 The scheduler shall have a set of pre-defined resource (number of nodes, memory, storage) usage thresholds per job, workflow or set of workflows;
- R4 The proposed scheduler shall allow a user to override predefined thresholds;
- R5 The proposed scheduler shall lead to auditable performance changes while saving energy;
- R6 The proposed scheduler shall be able to handle multiple workflows in parallel;
- R7 The proposed scheduler shall be fault tolerant and handle workflow failure gracefully;
- R8 All data generated by the scheduler shall be logged for debugging;
- R9 The proposed scheduler shall allow export of data;
- R10 The proposed scheduler shall be compatible across different platforms.

B. Challenges for the development of an Energy-Aware Scheduler

The following challenges were identified during the literature review that affect the development of the proposed scheduler.

- **A workflow management is a complex process.** Workflow Management Systems perform a number tasks such as executing, logging, pausing, resuming, etc. of workflows. These are independent tasks that handle different aspects of the workflow management. The scheduler needs to be able to understand the workings and execution of the workflows. It shall then take decisions based on all the factors.
- **Multiple Workflow Management Systems (WMS) are available.** Different workflow management systems like Apache Airavata, Kepler, Apache Airflow and Pegasus have been developed with different use cases and features. The scheduler can be developed to work with one of the management systems or can work independently.
- **Different parallel message passing interfaces.** Depending on the workflow, a number of different message

passing interfaces such as mpich, MPI, mpi4py are used for inter node communication. The scheduler can be developed to work with one of them or can work independently.

- **Collection of accurate energy data.** Energy consumption data collected from the sensors are not very precise due to sensor error margins and accessibility issues.
- **Multiple workflows with different parameters.** Workflows with varying parameters and structure makes it hard to develop a generic scheduler that can take into consideration all of these parameters and take decisions based on them.
- **Resource Contention.** Different workflows can be scheduled on a pool of resources. This is common practise now a days and can lead to resource contention that can affect the energy cost of a workflow. Attempting to reduce the energy of a workflow in such scenario is a complex task depending on number of external and internal factors of the workflow and the computing resources.

IV. Energy-aware scheduling

The previous sections presented the requirements and challenges for an energy aware scheduler. In this section, a generic high level design for an energy aware scheduler is presented. The design follows and aims to meet all the user requirements outlined in Section III.

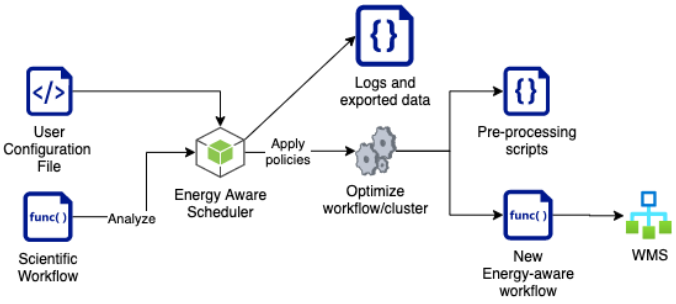

Fig. 3. Energy Aware Scheduler

The high level working of the scheduler can be described as shown in Figure 3. The design presented works independently to generate new energy aware workflows and configure cluster according to a set of policies. This new workflow can then be executed using the existing Workflow Management Systems (WMS). The design requires minimal installation for its setup and working.

In response to the requirements set out in Section III, the core working of the scheduler can be categorised into three main components as shown in Figure 4. The scheduler can understand the workflow, it's execution, dependencies, the computing environment, etc. and can configure the workflow and the cluster in order to maximise the cluster utilization and reduce the energy consumption. The scheduler analyzes the workflow and develops an energy efficient solution using this three part process.

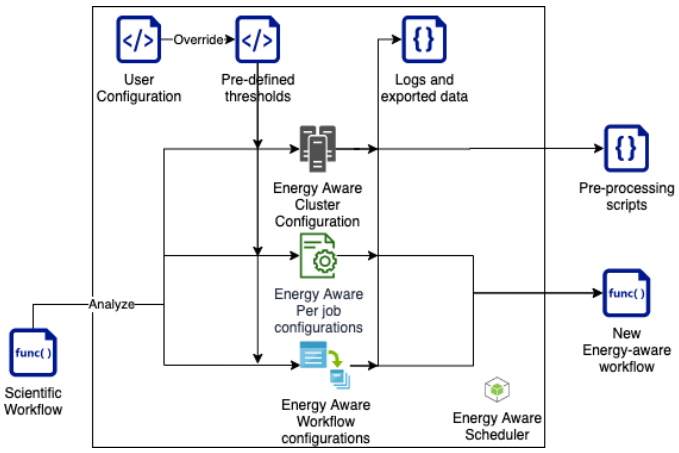

Fig. 4. Detailed breakdown of the energy aware scheduler.

A set of predefined thresholds are provided to the scheduler which relate to the user's need and workflow characteristics. A user configuration file can be defined which over-rides the default thresholds. These functionalities are in response to R3 and R4. As per R9 and R10, the scheduler is developed in a platform-independent universal programming language that and is capable of generating logs files which can be exported for further analysis.

The scheduler does not change the workflow or the data used for computation. Consequently, the scheduler does not vary the output of the workflow during multiple execution of the same workflow as outlined in R2. Also, in response to R7, the scheduler generates a new energy efficient workflow that is executed using a WMS. This workflow is tested for breaks or faults before submitting it for execution. The three part process shown in Figure 4 is further discussed in this section.

A. Cluster energy-aware scheduling (e.g. parallel workflows)

This section presents the optimization that the scheduler can perform on the cluster to make it more energy efficient and optimal for a particular workflow. These optimizations act as policies for the scheduler and the scheduler makes decisions based on one or more of these policies.

- **Switching off the nodes that may not be used.** The scheduler analyzes the workflow and decides the maximum number of nodes the workflow might use based on the maximum number of parallel jobs. The nodes that are known to be idle during the whole execution of the workflow, i.e. surplus computing resources, can be shut down to save energy. In certain cases, when the node is initially used and then idle for long time, it can be dynamically turned off to save energy.
- **Turning on new nodes if the available resources are less than required.** During times of peak computation, if the available resources are insufficient then the scheduler can dynamically power a node on. This can only be

done when the performance increase compensates for the increase in energy consumption (R5).

- **Specifying the number of threads to be used on each node.** According to the complexity of the jobs in a workflow, the scheduler can specify the number of threads to use on certain nodes depending on the memory requirements. Computationally expensive tasks can benefit from more CPU and memory usage.
- **Optimizing the CPU frequency in the nodes.** Over-volting is performed to increase the energy consumption of the CPU in order to increase the computing power of CPU. This affects the energy consumption and performance of a node substantially. Computationally intensive tasks can benefit from over-volting as long as the performance increase is greater than the energy consumption. Similarly, under-volting is performed when less computationally intensive tasks such as data transfer, unzipping, zipping, etc are to be executed.
- **Changing network settings.** There are many different cluster setups used for high performance computing. They vary from one huge supercomputer to multiple small distributed computer networks across the globe. The scheduler can exploit the network settings to further improve the energy consumption of the whole cluster. For example, during network booting of nodes, changing the NFS block size can affect how fast the changes are synced. For distributed computing systems, different settings such as connection medium, Ethernet links, switch configurations, etc. can be altered to optimize the cluster for inter-communication of nodes.
- **Turning off non essential background processes and components on the nodes.** The scheduler can turn of all non essential processes and components on the node that consume energy and do not contribute to the computation. Background processes such as system logs, redundant processes from closed apps, etc. can consume a lot of energy and CPU memory usage. Components such as empty USB ports, LEDs, HDMI, AUX, etc. can be turned off to save energy consumption of a node.

B. Energy-Aware job scheduling

This section presents the optimization that the scheduler can perform on individual jobs in a workflow based on their characteristics such as pre-processing, data, network usage, CPU load, post-processing, etc. to make the workflow execute more efficiently.

- **Scheduling bottleneck jobs first.** The job priority determines which job gets queued before the others. If multiple jobs have same priorities then all the jobs get queued and are executed as the resources become available. Normally all same level jobs in a workflow are given same priority and this is an issue as one job could be a bottleneck and releasing it early can be beneficial for the overall execution of the workflow (R8). The scheduler understands the DAG of the workflow and identifies the bottlenecks. It can then assign higher priorities to the jobs that may benefit from executing early.
- **Targeting jobs at specific nodes.** In a hybrid cluster environment with a different mixture of OS's, processor architectures, memory allocations, etc. it is beneficial to target jobs to specific nodes based on their complexity. Data intensive jobs can benefit from non distributed cluster setups. Similarly, distributed nodes with huge memory and CPU power can be used to schedule CPU and memory intensive jobs. The scheduler makes use of different aspects of the tasks and can find the optimal scheduling technique to reduce the overall energy consumption of the computation.
- **Prioritise based on job size.** CPU intensive jobs are often paired with high memory requirement. This might not be the case always and the scheduler can identify these discrepancies. It can then edit the requirements of the jobs to executed them specifically on nodes that can execute the jobs effectively and with less energy usage.

C. Energy-Aware workflow scheduling

A workflow has many specific configurations that are specific to itself and can be exploited for an overall decrease in energy consumption and increase in performance. This sections presents the optimization that the scheduler can perform based on the workflow specific aspects.

- **Using local binaries on the nodes.** Many workflows require installation of workflow specific binaries that help in execution of the tasks. These binaries are platform dependent and different binaries are transferred to nodes with different architecture as a part of workflow execution. This substantially increases the data transfer overhead as these binaries are transferred every time a job is scheduled. The scheduler can analyze the workflow and decide to locally install the binaries on the nodes beforehand, thus saving lot of communication overhead (R8). This leads to huge performance gain and subsequently reduce energy usage.
- **Changing the scheduling algorithm.** Depending on the complexity of the workflow and bottlenecks, different scheduling algorithms such as round-robin and grasshopper can be used for effective scheduling.
- **Multiple workflow execution.** Multiple workflows can be executed in tandem on a resource pool. This can lead to huge queuing and resource contention as different jobs compete for the available resources. The scheduler can identify the independent and bottleneck jobs in multiple workflows and execute them first for smooth flow of the workflow. It can also intelligently queue jobs on nodes so as to reduce the idle time of the nodes while waiting for job dependency to be fulfilled (R6).

V. Conclusion and Future Work

Software development for scientific workflow execution on clusters has mainly focused on improving run-time perfor-

mance. In this paper, we argue for an energy-aware scheduler for scientific computing to address this shortcoming.

This paper proposes a scheduler that utilises cluster resources in an energy efficient way for executing workflows. Three different aspects of optimizing computation are discussed which provide a structure on which the scheduler is to be developed. The scheduler makes use of different policies to optimize the workflow execution. It takes into consideration all the aspects of a workflow execution, from optimizing the cluster, to scheduling jobs on specific nodes, to reducing resource contention among multiple workflow executions.

In future work, the approach and design is to be formalised and the scheduler implementation evaluated on a number of real-world scientific workflows. Energy-Aware workflow execution will be evaluated against normal workflow execution and the performance of the scheduler demonstrated.

REFERENCES

[1] I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields *et al.*, *Workflows for e-Science: Scientific Workflows for Grids*, 1st ed. Springer, London: Springer, December 2007, vol. 1.

[2] M. Warade, J.-G. Schneider, and K. Lee, "Fepac: A framework for evaluating parallel algorithms on cluster architectures," in *2021 Australasian Computer Science Week Multiconference*, 2021, pp. 1–10.

[3] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.

[4] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services," *Nucleic acids research*, vol. 34, no. suppl_2, pp. W729–W732, 2006.

[5] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system," *Concurrency and computation: Practice and experience*, vol. 18, no. 10, pp. 1039–1065, 2006.

[6] J. C. Sloan, T. M. Khoshgoftaar, and V. Raghav, "Assuring timeliness in an e-science service-oriented architecture," *Computer*, vol. 41, no. 8, pp. 56–62, 2008.

[7] Q. Wu and V. V. Datla, "On performance modeling and prediction in support of scientific workflow optimization," in *2011 IEEE World Congress on Services*. IEEE, 2011, pp. 161–168.

[8] J. Kim, E. Deelman, Y. Gil, G. Mehta, and V. Ratnakar, "Provenance trails in the wings/pegasus system," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 5, pp. 587–597, 2008.

[9] J. Li, Y. Fan, and M. Zhou, "Performance modeling and analysis of workflow," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 34, no. 2, pp. 229–242, 2004.

[10] M. Warade, J.-G. Schneider, and K. Lee, "Measuring the energy and performance of scientific workflows on low-power clusters," *Electronics*, vol. 11, no. 11, p. 1801, 2022.

[11] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Joint computation offloading and scheduling optimization of iot applications in fog networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 3266–3278, 2020.

[12] S. Giampà, L. Belcastro, F. Marozzo, D. Talia, and P. Trunfio, "A data-aware scheduling strategy for executing large-scale distributed workflows," *IEEE Access*, vol. 9, pp. 47 354–47 364, 2021.

[13] T. Saillant, J.-C. Weill, and M. Mougeot, "Predicting job power consumption based on rjms submission data in hpc systems," in *International Conference on High Performance Computing*. Springer, 2020, pp. 63–82.

[14] M. F. Cloutier, C. Paradis, and V. M. Weaver, "A raspberry pi cluster instrumented for fine-grained power measurement," *Electronics*, vol. 5, no. 4, p. 61, 2016.

[15] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, 2013.

[16] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M.-H. Su, "Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand," in *Optimizing Scientific Return for Astronomy through Information Technologies*, vol. 5493, 2004, pp. 221–232.

[17] J. C. Jacob, D. S. Katz, G. B. Berriman, J. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, T. A. Prince *et al.*, "Montage: An astronomical image mosaicking toolkit," *Astrophysics Source Code Library*, pp. ascl–1010, 2010.

[18] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling, "Scientific workflow applications on amazon ec2," in *2009 5th IEEE international conference on e-science workshops*. IEEE, 2009, pp. 59–66.

[19] L. Clarke, S. Fairley, X. Zheng-Bradley, I. Streeter, E. Perry, E. Lowy, A.-M. Tassé, and P. Flicek, "The international Genome sample resource (IGSR): A worldwide collection of genome variation incorporating the 1000 Genomes Project data," *Nucleic Acids Research*, vol. 45, no. D1, pp. D854–D859, 09 2016.

[20] 1000 Genomes Project Consortium, "A global reference for human genetic variation," *Nature*, vol. 526, no. 7571, p. 68, 2015.

[21] W. McLaren, L. Gil, S. E. Hunt, H. S. Riat, G. R. Ritchie, A. Thormann, P. Flicek, and F. Cunningham, "The ensembl variant effect predictor," *Genome biology*, vol. 17, no. 1, pp. 1–14, 2016.

[22] K. Lee, N. W. Paton, R. Sakellariou, E. Deelman, A. A. Fernandes, and G. Mehta, "Adaptive workflow processing and execution in pegasus," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 16, pp. 1965–1981, 2009.

[23] K. Lee, N. W. Paton, R. Sakellariou, and A. A. Fernandes, "Utility driven adaptive workflow execution," in *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE, 2009, pp. 220–227.

[24] K. Lee, N. W. Paton, R. Sakellariou, and A. Fernandes, "Utility functions for adaptively executing concurrent workflows," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 6, pp. 646–666, 2011.

[25] I. Pietri, M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, and R. Sakellariou, "Energy-constrained provisioning for scientific workflow ensembles," in *2013 International Conference on Cloud and Green Computing*. IEEE, 2013, pp. 34–41.

[26] T. Thanavanich and P. Uthayopas, "Efficient energy aware task scheduling for parallel workflow tasks on hybrids cloud environment," in *2013 International Computer Science and Engineering Conference (ICSEC)*. IEEE, 2013, pp. 37–42.

[27] J. Bader, L. Thamsen, S. Kulagina, J. Will, H. Meyerhenke, and O. Kao, "Tarema: Adaptive resource allocation for scalable scientific workflows in heterogeneous clusters," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 65–75.

[28] T. M. Sardaraz Muhammad, "A hybrid algorithm for scheduling scientific workflows in cloud computing," *IEEE Access*, vol. 7, pp. 186 137–186 146, 2019.

[29] Sardaraz and Tahir, "A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing," *International Journal of Distributed Sensor Networks*, vol. 16, no. 8, 2020.

[30] I. Pietri and R. Sakellariou, "Energy-aware workflow scheduling using frequency scaling," in *Proceedings of 43rd International Conference on Parallel Processing Workshops*. IEEE, 2014, pp. 104–113.

[31] F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 257–271, 2018.

[32] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai, "An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 455–480, 2020.

[33] E. N. Watanabe, P. P. Campos, K. R. Braghetto, and D. M. Batista, "Energy saving algorithms for workflow scheduling in cloud computing," in *2014 Brazilian Symposium on Computer Networks and Distributed Systems*. IEEE, 2014, pp. 9–16.

[34] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2018.