JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 40, 957-977 (2024)
DOI: 10.6688/JISE.202409_40(5).0002

Energy and Scientific Workflows: Smart Scheduling and Execution

MEHUL WARADE[1,+], KEVIN LEE[1], CHATHURIKA RANAWEERA[1] AND JEAN-GUY SCHNEIDER[2]
[1]*School of Information Technology*
Deakin University
Geelong VIC 3220, Australia
E-mail: mehul.warade@research.deakin.edu.au
[2]*Faculty of Information Technology*
Monash University
Clayton VIC 3168, Australia

Energy-efficient computation is an increasingly important target in modern-day computing. Scientific computation is conducted 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. Evidently, minimizing both execution time as well as energy consumption does not have to be mutually exclusive. The aim of the research presented in this paper is to highlight the benefits of energy-aware scientific workflow execution. In this paper, a set of requirements for an energy-aware scheduler are outlined and a conceptual architecture for the scheduler is presented. The evaluation of the conceptual architecture was performed by developing a proof of concept scheduler which was able to achieve around 49.97% reduction in the energy consumption of the computation.

Keywords: energy-aware computing, scientific workflows, scheduling, high performance computing, parallel computing

1. INTRODUCTION

A scientific workflow is defined as a 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 energy consumption has been seen for multiple workflows [2]. This often means that after a certain time, using more cluster resources results in minimal computational improvements but large energy overheads. Workflow engines 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 handle the creation, management, and debugging of the cluster and workflow. This makes it difficult for them to be able to understand the

Received December 19, 2022; revised June 2, 2023; accepted June 30, 2023.
Communicated by Huo Chong Ling.
[+] Corresponding author.

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 modeling and analyzing 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 analyzing 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 the 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. A proof-of-concept scheduler is developed based on the conceptual architecture and is evaluated by executing a scientific workflow and comparing the execution with the standard execution. The authors anticipate that a formal scheduler built to address the requirements and policies 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 energy-aware schedulers; (iii) a conceptual architecture for an energy-aware scheduler for scientific workflows; and (iv) a proof of concept implementation and evaluation of the proposed scheduler. This paper is an extended version of a conference paper [13], which includes new experimental results, an evaluation of the proposed solution, and additional discussion.

The remainder of this paper is as follows: Section 2 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 3. Section 4 presents a conceptual architecture for developing an energy-aware scheduler. The evaluation of conceptual architecture by executing a scientific workflow on a cluster is presented in Section 5. Finally, Section 6 provides conclusions and future work.

2. BACKGROUND

High-Performance Computing (HPC) has led to an increase in global energy usage [14]. 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 energy usage and developing more energy-efficient methods for computing is of ever-increasing importance [14, 15]. 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.

2.1 Scientific Workflows

Workflows are increasingly being used in High-Performance Computing (HPC) to accommodate for the growing complexity of computation, simulations, and analysis [16]. A workflow comprises of a number of individual jobs that are executed 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.

2.1.1 Montage

Montage [17, 18] 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 [19] compared to other scientific workflows. The Montage toolkit can be used to generate workflows of varying sizes 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 color channels which the final images should be generated from. Fig. 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).

2.1.2 Bioinformatics

The Bioinformatics workflow is based on the data collected by the 1,000 Genomes Project [20, 21]. 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 mutation's sift scores (calculated by the Variant Effect Predictor [22]) can help researchers in discovering the exact mutation which is the cause of 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 the workflow (the data to be computed) and the number of parallel jobs. Fig. 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 is computationally more intensive than others and the bio-informatics workflow has 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 the workflow and the number of cluster nodes has a direct impact on the queuing of jobs, execution time, and energy consumption of the workflow. Identifying such jobs and smartly executing them can lead to improvements in energy consumption and the performance of the workflows.

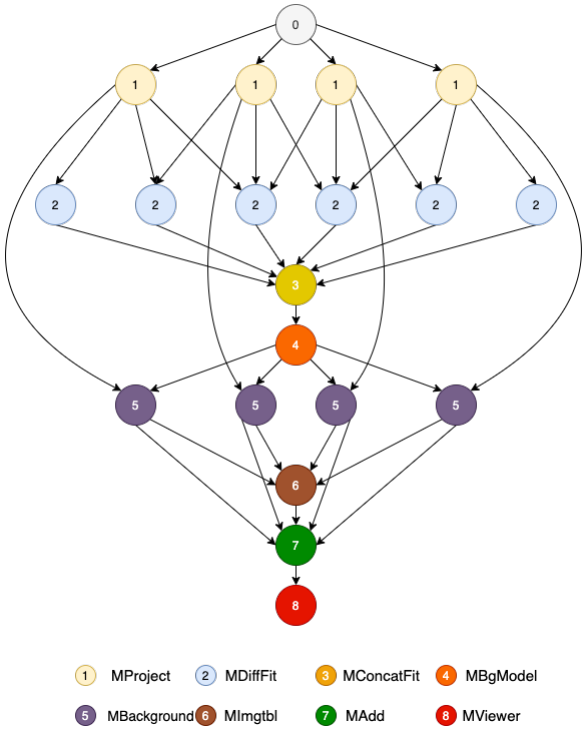

Fig. 1. A simple Montage workflow.

2.2 Workflow Schedulers

Optimizing schedulers for workflows have always been an area of interest for scientists. An adaptive workflow processing and execution method are possible [23–25]. 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 [26]. 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 the performance of workflows in a cloud-based distributed systems [27].

A system for dynamically allocating tasks based on the available infrastructure and knowledge of the workflow has been developed [28]. 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 a reduction of 66% in execution time over traditional schedulers. A hybrid algorithm making use of the Particle swarm optimization (PSO) algorithm and processing bottleneck tasks on high priority has been developed to achieve better execution time along with no loss in cluster load [29]. Similar results have been obtained from a scheduler developed by using genetic algorithm [30].

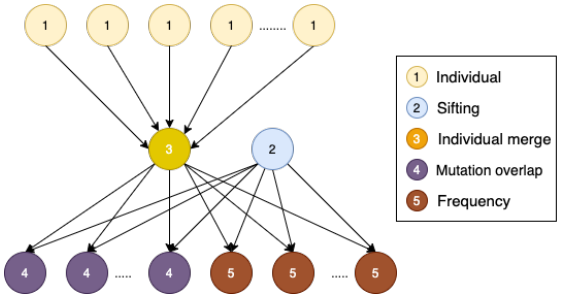

Fig. 2. A simple bioinformatics workflow.

2.3 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 into consideration while scheduling the workflows.

Scheduling algorithms have also been introduced to meet time deadlines of computation while minimizing the energy consumption [9,31]. A scheduler based on a polynomial time algorithm is proposed to provide real-time dynamic resource allocation in order to get good solutions in a particular time [32]. 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 [33].

Cloud computing is one of the most researched areas for HPC and energy savings. Not everyone can afford huge data centers and, cloud computing is the go-to for any researcher looking for computation. Inter-dependency of tasks leads to huge data transfer and fewer computation tasks being executed in the cloud. A scheduler to reduce the data transfer and inter-dependency have been developed with the focus on reducing the energy footprint of the workflow [34]. The scheduler was able to achieve a 22.7% reduction in energy at no cost to the makespan 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 the Lyapunov optimization technique to minimize the queuing of tasks and achieves an 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 [35]. This policy dynamically optimizes the CPU clock frequency and the wireless transmission power to achieve a reduction in the energy-efficiency cost (EEC) of the workflow.

The current research in the domain of energy-aware scheduling is workflow specific and cannot be used for other computations in the scientific community. These schedulers do not take into account the optimizations that can be made on workflow, job, and cluster levels. 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.

3. REQUIREMENTS AND CHALLENGES FOR AN ENERGY-AWARE SCHEDULER

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

3.1 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 have been 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 the export of data;

R10 The proposed scheduler shall be compatible across different platforms.

3.2 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 of tasks such as executing, logging, pausing, resuming, *etc*. of workflows. These are independent tasks that handle different aspects of 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, and 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 structures make 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 a common practice nowadays 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 a scenario is a complex task depending on the number of external and internal factors of the workflow and the computing resources.

4. ENERGY-AWARE SCHEDULING

The previous sections presented the requirements and challenges for an energy-aware scheduler. In this section, a generic high-level conceptual design for an energy-aware scheduler is presented. The design follows and aims to meet all the user requirements outlined in Section 3.

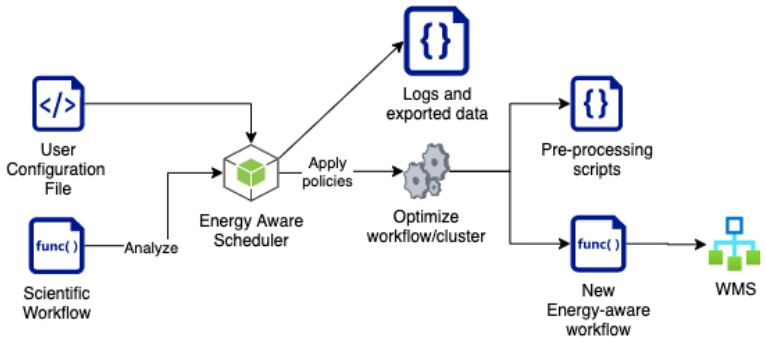

Fig. 3. Energy aware scheduler.

The high-level working of the scheduler can be described as shown in Fig. 3. The design presented works independently to generate new energy-aware workflows and configure clusters 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.

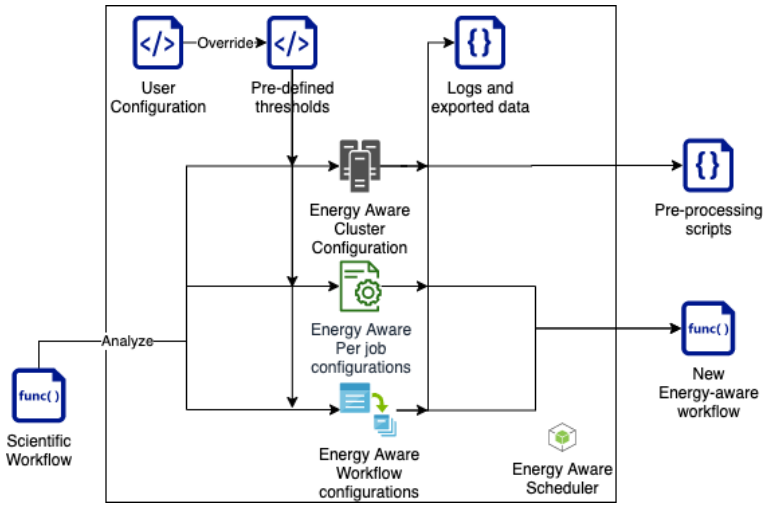

Fig. 4. Detailed breakdown of the energy-aware scheduler.

In response to the requirements set out in Section 3, the core working of the scheduler can be categorized into three main components as shown in Fig. 4. The scheduler can understand the workflow, its execution, dependencies, the computing environment, *etc*., and can configure the workflow and the cluster in order to maximize cluster utilization and reduce energy consumption. The scheduler analyzes the workflow and develops an energy-efficient solution using this three-part process.

A set of predefined thresholds are provided to the scheduler which relates to the user's need and workflow characteristics. A user configuration file can be defined which overrides 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 is capable of generating log files that can be exported for further analysis.

The scheduler does not change the original workflow or the data used for computation. Consequently, the scheduler does not vary the output of the workflow during multiple executions 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 Fig. 4 is further discussed in this section.

4.1 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 a 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 the CPU. This affects the energy consumption and performance of a node substantially. Computationally intensive tasks can benefit from overvolting as long as the performance increase is greater than the energy consumption. Similarly, undervolting 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 the 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 the inter-communication of nodes.

- **Turning off non-essential background processes and components on the nodes.** The scheduler can turn off all non-essential processes and components on the node that consumes 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 the energy consumption of a node.

4.2 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 the 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 the 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 OSs, 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 requirements. This might not be the case always and the scheduler can identify these discrepancies. It can then edit the requirements of the jobs to execute them specifically on nodes that can execute the jobs effectively and with less energy usage.

4.3 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 an increase in performance. This section presents the optimization that the scheduler can perform based on workflow-specific aspects.

- **Using local binaries on the nodes.** Many workflows require the installation of workflow-specific binaries that help in the execution of the tasks. These binaries are platform-dependent and different binaries are transferred to nodes with different architectures 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 a 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 a smooth workflow flow. 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).

5. EVALUATION

Section 4 describes the generic conceptual design of an Energy-Aware Scheduler based on the requirements set out in Section 3. The scheduler introduces a solution to reduce the energy consumption of the computation by tackling its three main parts – whole cluster, individual jobs, and the workflow dependencies [13]. This three-point solution focuses on the major aspects of workflow execution that can affect energy consumption and aids the scheduler in the decision-making process. To confirm the functioning of the proposed scheduler, a proof of concept scheduler is developed and evaluated on a scientific workflow in this section.

In order to have a better representation of a real-world data center with different types of computer hardware and resources available for different computations, a heterogeneous cluster of 10 single-board computers (SBCs) of different configurations was developed. Single Board Computers were chosen as they have been identified as energy efficient for the amount of performance that they provide [2, 10, 36].

5.1 Experimental Setup

Fig. 5 presents the cluster developed to perform computations for this study. The 10 nodes included five Raspberry Pi 4B (Node IDs – 1 to 5) and five Raspberry Pi 3B+ (Node IDs - 6 to 10). Raspberry Pi 4B (RPi 4B) have 2GB RAM as compared to 1GB RAM for the Raspberry Pi 3B+ (RPi 3B+). The processor in the RPi4B is the Broadcom BCM2711 Quad-core ARM Cortex-A72 and RPi3B+ has a Broadcom BCM2837 Quad-core ARM Cortex-A53 processor. All the nodes are connected to each other and the master node by 2 Netgear GS110TP switches. These switches provide power to the Raspberry Pi's through Power over Ethernet (PoE). The switch then provides easy access to the energy consumption data of these nodes via its internal power monitoring sensors. The 10-node cluster was managed by an x86 Linux-based Intel NUC (4 core 8 threads with Linux Mint OS (`https://linuxmint.com/`, accessed on 7 October 2022) that acted as a master node.

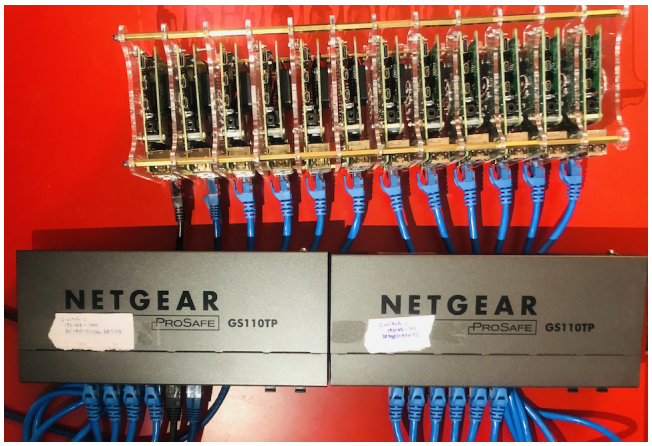

Fig. 5. Cluster

The scheduler is evaluated using the execution of a 10k data variant of the Bioinformatics Workflow (See 2.1) on the cluster. The workflow is executed using the Pegasus Workflow Management System (WMS). Pegasus is responsible for the submission, execution, and management of the workflow [3]. Pegasus makes use of HTCondor, a scheduler to submit, schedule and execute jobs [37]. The scheduler proposed in this paper alters the requirements for the executing jobs and manipulates the HTCondor to schedule them according to the policies on the cluster. RPis are quad-core in nature and as HTCondor considers each core as an independent resource, the maximum number of jobs that a single RPi can execute at any time is 4. In total, the cluster developed in this study can execute 40 jobs in parallel.

The standard execution of the workflow is compared with the execution of two policies that the scheduler used to schedule the jobs and the performance and energy consumption of these executions are collected and analyzed. The amount of resources being used (see Figs. 6, 8 and 10) and the total number of jobs being executed on the cluster (see Figs. 7, 9 and 11) as compared to the energy consumption of the cluster are the two major metrics used to compare the performance of the scheduler. The jobs that are being executed on the cluster are considered in the analysis. The jobs being conducted by the master nodes are not considered as they are not related to the computation and are just majorly related to file transfers or folder creation.

5.2 Normal/ Standard Execution

Standard execution of the Bioinformatics Workflow includes creating the workflow using Pegasus and submitting it to the cluster. There are no extra configurations associated with executing the workflow. All the default and standard configuration options are used. This experiment denotes the normal workflow execution that any scientist performs using Pegasus. The data from this experiment is considered a baseline for any future comparisons between the different policies of the scheduler.

The execution data of the experiment is shown below. Fig. 6 illustrates the number of active threads on each node during the execution of the workflow. The X-axis indicates the execution time (in seconds) of the workflow at any given instant and the Y-axis indicates the node ID. Different usage of nodes is denoted by different coloured dots. The analysis only considers the jobs that are executed on the cluster nodes.

Fig. 7 illustrates the breakdown of the jobs that are being executed on the cluster and the energy consumption of the cluster. The X-axis indicates the execution time (in seconds) of the workflow at any given instant. The left Y-axis is the instantaneous energy consumption of the cluster and the right Y-axis denotes the number of jobs for each job type being executed on the cluster. Different job types are denoted by different colors.

The standard execution of the Bioinformatics workflow on the heterogeneous cluster of 5 RPi 4B and 5 RPi 3B+ took 6,658 seconds (approximately 1 hr and 51 min). The cluster uses the maximum energy during the execution of the *individuals* job. As can be seen from Figs. 6 and 7, the majority of the cluster is idle during the execution of *individuals_merge* job. This is expected behavior as the *indiviual_merge* job is a bottleneck in the Bioinformatics workflow (see Fig. 2). The execution of the workflow has to completely stop and cannot proceed without the completion of this job. The *individuals* job is compute-intensive and a drop in energy consumption can be as soon as the job is complete

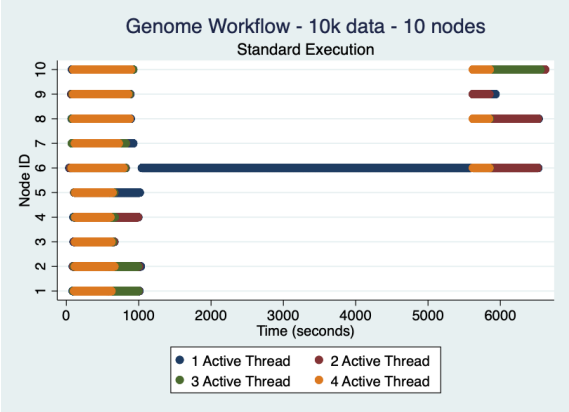

Fig. 6. Number of Active Threads per node for a bioinformatics workflow for standard execution.

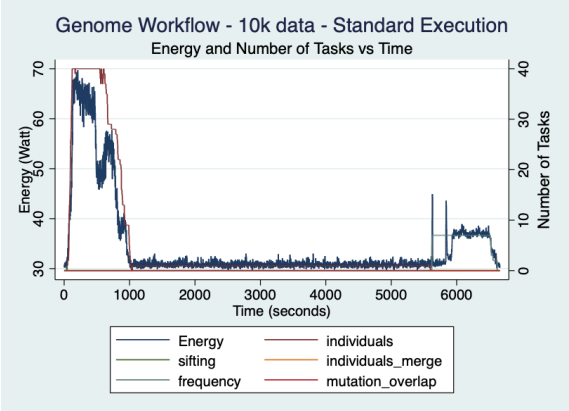

Fig. 7. Number of tasks and Energy Consumption over workflow execution time for standard workflow execution.

and other jobs are executed. Considering that this experiment denotes the standard execution of a workflow, the idle nodes still consume their base energy during the execution. The standard execution of the Bioinformatics workflows consumed around 63.92 Watt-hr of energy.

It is important to note that the selection of the node to execute the bottleneck job is dependent on the internal scheduling policies of the HTCondor [37]. HTCondor schedules jobs based on the requirement of the job and the available resources. In this instance, all the nodes can execute the job but only node 6 (see Fig. 6) was available at that instant which resulted in HTCondor scheduling the job on those nodes.

The execution time and energy consumption data of each individual job are used to develop policies that will help the scheduler in achieving the reduction in energy consumption and improvement of performance in the computation. For example, *individuals* job on the RPi 4B took around 507 seconds (approximately 8 min and 27 sec) and RPi

3B+ took around 749 seconds (approximately 12 min and 29 sec). Similar data were collected for all the different jobs and used to develop policies for the scheduler accordingly. The data found that RPi 4B (Nodes 1-5) performs better than RPi 3B+ (Nodes 6-10) in the majority of the computations.

5.3 Energy Aware Scheduling – Scheduling a Single Set of Jobs on RPi 3b+

The previous section presented the standard execution data of the Bioinformatics workflow. A proof of concept scheduler based on the conceptual architecture presented in this paper (see Section 4) was developed and the Bioinformatics workflow is scheduled and executed according to the policies set by the scheduler. The main aim of the scheduler is to achieve a reduction in the energy consumption of the workflow at no or minimal cost to the performance. A policy to utilize the whole cluster to execute the compute-intensive jobs and then schedule the remaining jobs on nodes that have better performance is being evaluated in this section.

Fig. 8 illustrates the number of active threads on each node during the execution of the workflow. The X-axis indicates the execution time (in seconds) of the workflow at any given instant and the Y-axis indicates the node ID. Different usage of nodes is denoted by different coloured dots.

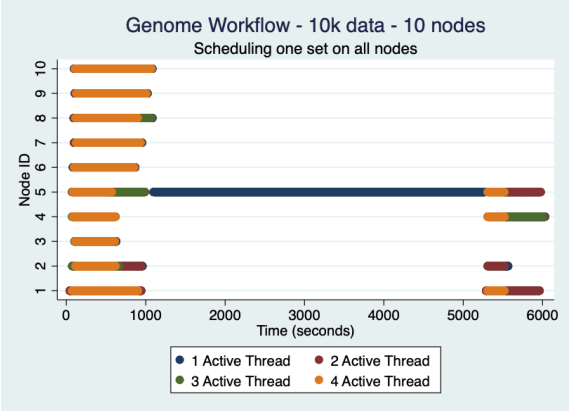

Fig. 8. Number of Active Threads per node for a bioinformatics workflow for scheduling a single set of jobs on all nodes.

Fig. 9 illustrates the breakdown of the jobs that are being executed on the cluster and the energy consumption of the cluster. The X-axis indicates the execution time (in seconds) of the workflow at any given instant. The left Y-axis is the instantaneous energy consumption of the cluster and the right Y-axis denotes the total number of jobs being executed on the cluster. Different jobs are denoted by different colours.

The scheduler used the following policy for the execution – after the initial execution of compute-intensive jobs on all the nodes, all the remaining jobs are scheduled only on the nodes with higher performance. This can be confirmed from Fig. 8 that illustrates the scheduled tasks on the nodes. A similar observation to that of the standard execution can be seen while scheduling the bottleneck job (scheduled on node 5). The total execution

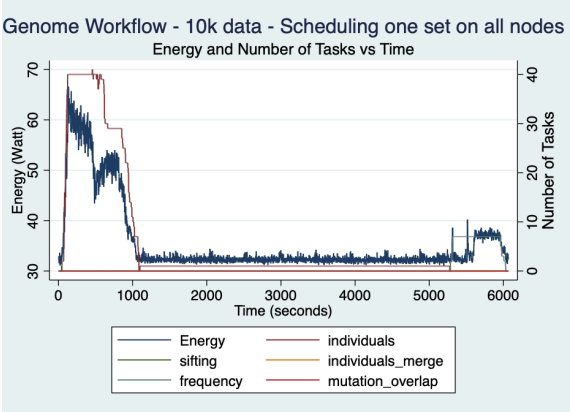

Fig. 9. Number of tasks and Energy Consumption over workflow execution time for scheduling a single set of jobs on all nodes.

time of the workflow was 6,067 seconds (approximately 1 hr and 41 min), an improvement of around 8.86% in the performance of the workflow as compared to that of standard execution. The energy consumption of the execution is found to be 38.25 Watt-hr (40.26% reduction in energy consumption as compared to that of standard execution). As presented in Section 4, the scheduler powers the idle nodes off.

5.4 Energy Aware Scheduling – Scheduling All Jobs on RPi4B

The previous experiment showed that the RPi 4B is very energy efficient as compared to the RPi 3B+. To investigate this further, the scheduler was used to execute the same workflow again with a different policy. The policy was to schedule the jobs only on the RPi 4B and to consider the RPi 3B+ as powered off and not in use.

The execution data of the experiment is shown below. Fig. 10 illustrates the number of active threads on each node during the execution of the workflow. The X-axis indicates the execution time (in seconds) of the workflow at any given instant and the Y-axis indicates the node ID. Different usage of nodes is denoted by different colored dots. The analysis only considers the job that is executed on the cluster nodes.

Fig. 11 illustrates the breakdown of the jobs that are being executed on the cluster and the energy consumption of the cluster. The X-axis indicates the execution time (in seconds) of the workflow at any given instant. The left Y-axis is the instantaneous energy consumption of the cluster and the right Y-axis denotes the total number of jobs being executed on the cluster. Different jobs are denoted by different colours.

As the number of nodes that are being used is only 5, the maximum number of jobs that can be parallelly executed is 20. This can be confirmed from Fig. 11. A similar observation to that of the standard execution can be seen while scheduling the bottleneck job (scheduled on node 4). The maximum number of the *individuals* jobs that are being executed parallelly is 20. Due to this, the max energy consumption of the whole cluster is also around 35 Watts as compared to 65 Watts in the previous experiments. The total execution time of the workflow was 6,305 seconds (approximately 1 hr and 45 min). The

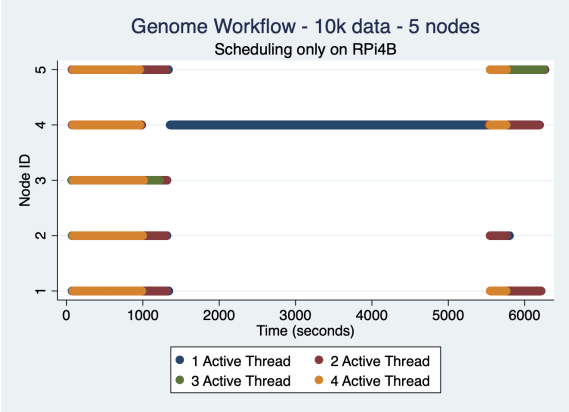

Fig. 10. Number of Active Threads per node for a bioinformatics workflow for scheduling all on RPi4B.

execution time is 5.29% less than that of the standard execution but 3.92% higher as compared to the previous policy.

The total energy consumption of the computation in this policy was around 32.23 Watt-hr. This is a decrease of around 49.97% as compared to the standard execution (see Section 5.2). Comparing this with the execution data from the previous policy, this policy achieved a reduction of 15.74% in energy consumption. In terms of the execution time, the policy to schedule all the jobs on RPi 4B performs poorly than the policy to schedule compute-intensive jobs on all nodes and then schedule all remaining jobs on high-performing nodes. When comparing energy consumption, the current policy performs better than the standard execution and the execution using the previous policy. The debatable question here can be if the reduction of energy consumption is worth the increase in execution time.

6. CONCLUSION AND FUTURE WORK

Software development for scientific workflow execution on clusters has mainly focused on improving run-time performance. In this paper, we argue for an energy-aware scheduler for scientific computing to address this shortcoming. The Scheduler design described in Section 4 takes into account all the attributes of the workflow/ cluster. Based on the data, the scheduler is able to configure the cluster and schedule the workflow in an energy-efficient manner. The scheduler requires minimal domain understanding and can understand and schedule the workflow totally based on the jobs, interdependencies, and complexity.

This paper proposed a scheduler that utilizes 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 workflow execution. The scheduler was able to achieve increased performance and reduced energy consumption of the workflow exe-

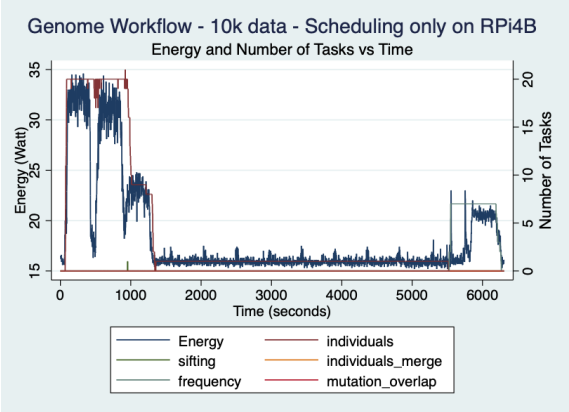

Fig. 11. Number of tasks and Energy Consumption over workflow execution time for scheduling all on RPi4B.

cution as compared to its standard execution.

The approach and experiments shown in this paper demonstrate that different scheduling policies can have a huge impact on the energy consumption and performance of workflow execution. In future work, the approach and design are to be formalized 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. This will enable scientists to be more energy aware of their computation and they can implement policies to reduce energy consumption or to optimize the performance of their computation.

ACKNOWLEDGMENT

Thanks to the GECOST Team for considering the conference paper for this extended journal.

REFERENCES

1. I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields *et al.*, *Workflows for e-Science: Scientific Workflows for Grids*, 1st ed., Vol. 1, Springer, London, 2007.
2. M. V. Warade, J.-G. Schneider, and K. Lee, "FEPAC: A framework for evaluating parallel algorithms on cluster architectures," in *Proceedings of 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, 2005, pp. 219-237.

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, 2006, pp. W729-W732.
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, 2006, pp. 1039-1065.
6. J. C. Sloan, T. M. Khoshgoftaar, and V. Raghav, "Assuring timeliness in an e-science service-oriented architecture," *Computer*, Vol. 41, 2008, pp. 56-62.
7. Q. Wu and V. V. Datla, "On performance modeling and prediction in support of scientific workflow optimization," in *Proceedings of IEEE World Congress on Services*, 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, 2008, pp. 587-597.
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, 2004, pp. 229-242.
10. M. Warade, J.-G. Schneider, and K. Lee, "Measuring the energy and performance of scientific workflows on low-power clusters," *Electronics*, Vol. 11, 2022, p. 1801.
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, 2020, pp. 3266-3278.
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, 2021, pp. 47354-47364.
13. M. Warade, J.-G. Schneider, and K. Lee, "Towards energy-aware scheduling of scientific workflows," in *Proceedings of International Conference of Green Energy, Computing and Sustainable Technology*, 2022, pp. 93-98.
14. 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*, 2020, pp. 63-82.
15. M. F. Cloutier, C. Paradis, and V. M. Weaver, "A raspberry pi cluster instrumented for fine-grained power measurement," *Electronics*, Vol. 5, 2016, p. 61.
16. G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, Vol. 29, 2013, pp. 682-692.
17. 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.
18. 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*, 2010, p. ascl-1010.

19. G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling, "Scientific workflow applications on amazon ec2," in *Proceedings of the 5th IEEE International Conference on e-Science Workshops*, 2009, pp. 59-66.
20. 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, 2016, pp. D854-D859.
21. 1000 Genomes Project Consortium, "A global reference for human genetic variation," *Nature*, Vol. 526, 2015, p. 68.
22. 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, 2016, pp. 1-14.
23. 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, 2009, pp. 1965-1981.
24. K. Lee, N. W. Paton, R. Sakellariou, and A. A. Fernandes, "Utility driven adaptive workflow execution," in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 220-227.
25. 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, 2011, pp. 646-666.
26. I. Pietri, M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, and R. Sakellariou, "Energy-constrained provisioning for scientific workflow ensembles," in *Proceedings of IEEE International Conference on Cloud and Green Computing*, 2013, pp. 34-41.
27. T. Thanavanich and P. Uthayopas, "Efficient energy aware task scheduling for parallel workflow tasks on hybrids cloud environment," in *Proceedings of IEEE International Computer Science and Engineering Conference*, 2013, pp. 37-42.
28. 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 *Proceedings of IEEE International Conference on Big Data*, 2021, pp. 65-75.
29. M. Sardaraz and M. Tahir, "A hybrid algorithm for scheduling scientific workflows in cloud computing," *IEEE Access*, Vol. 7, 2019, pp. 186137-186146.
30. M. Sardaraz and M. Tahir, "A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing," *International Journal of Distributed Sensor Networks*, Vol. 16, 2020.
31. I. Pietri and R. Sakellariou, "Energy-aware workflow scheduling using frequency scaling," in *Proceedings of IEEE 43rd International Conference on Parallel Processing Workshops*, 2014, pp. 104-113.
32. 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, 2018, pp. 257-271.
33. 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, 2020, pp. 455-480.
34. E. N. Watanabe, P. P. Campos, K. R. Braghetto, and D. M. Batista, "Energy saving algorithms for workflow scheduling in cloud computing," in *Proceedings of IEEE*

Brazilian Symposium on Computer Networks and Distributed Systems, 2014, pp. 9-16.
35. 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, 2018, pp. 319-333.
36. J. Liu, K. Wang, and F. Chen, "Understanding energy efficiency of databases on single board computers for edge computing," in *Proceedings of IEEE 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2021, pp. 1-8.
37. M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor-a hunter of idle workstations," Department of Computer Sciences, University of Wisconsin-Madison, Technical Report, 1987.

Mehul Warade is an international student from India pursuing his Doctorate at Deakin University, Australia. He completed his Bachelor of Software Engineering with H1 first class Honors (2017-2021) and presented his first academic publication while still completing his studies. His current research focuses in the field of affordable and energy constraint high-performance computing in clusters. His research interests include IoT, distributed and cloud computing, robotics and green computing.

Kevin Lee is a Senior Lecturer in Software Engineering and Internet of Things (IoT) at the School of Information Technology in Deakin University, He has over 80 publications, with an H-Index of 15 and over 1400 citations. His research is focused on distributed systems and autonomic computing, and he has published in the areas of IoT, cloud computing, robotics, embedded systems, big data, P2P network monitoring and physiological computing. His current active research area is IoT and its integration with cloud computing in support of dynamic IoT applications.

Chathurika Ranaweera is a Senior Lecturer at the School of Information Technology, Deakin University, Australia. She received a B.Sc in Engineering and Ph.D. degrees from The University of Peradeniya, Sri Lanka and The University of Melbourne, Australia, respectively. Her research interests include optical transport networks for 5G and beyond, computation and communication convergence for 6G, secured and scalable networks for IoT, optical-wireless convergence, network optimisations, and energy aware computation and communication.

Jean-Guy Schneider is a Professor in Software Engineering at Deakin University and has more than 20 years experience in the Higher Education sector. He has an M.Sc. and Ph.D. in Computer Science and Applied Mathematics from the University of Berne, Switzerland. His research lies in the general area of reliable software systems and is positioned at the intersection of Software Engineering and Computer Science. More specifically, his research interests are in object-oriented and concurrent/ distributed/ service-oriented systems, programming languages, and the definition of formal approaches for component-based Software Engineering. He has more than 100 publications in internationally leading journals and conferences.