Energy Aware Adaptive Scheduling of Workflows

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.*
[2]*Faculty of Information Technology, Monash University, Clayton VIC 3168, Australia.*
mehul.warade@research.deakin.edu.au

***Abstract*—Scientific workflows consist of multi-step computational tasks executing in the form of data flow and task dependencies. These workflows are defined to be long running and fault tolerant. There is evidence of improving performance achieved through run-time adaptive changes made to the workflow execution. The aim of the work presented in this paper is to highlight the benefits that adaptive scheduling of scientific workflows have on the energy consumption of the computation. In this paper, an architecture for the implementation of an energy-aware adaptive scheduler is presented. The monitoring, analysis, planning and execution (MAPE) model from autonomic computing is used to propose a set of run-time modifications that will be used by the scheduler to improve the performance and energy consumption of the workflow.**

***Index Terms*—Cluster Computing, Energy-Aware, Workflows, Adaptive, MAPE**

I. Introduction

Autonomic Computing (AC) is a concept that aims to make adaptive decisions using high level policies [1]. Autonomic computing constantly checks for changes in the environment and adapts itself to unpredictable changes. It was developed to overcome the complexity of computing system management and to reduce the threat to further growth. An autonomic computing framework comprises of local and global control schemes. These schemes have sensors (for monitoring), effectors (for adjusting to environment), knowledge, and planner for run-time awareness. All these components work in tandem to help the computing system to be self and environmentally aware. This architecture is referred to as the Monitor-Analyze-Plan-Execute (MAPE) model [2].

Workflows have been increasingly used to execute complex workloads [3]–[5]. Workflows are collection of multiple smaller tasks that work together to achieve a complete compute goal. These individual tasks can make use of parallel computing to achieve higher performance. Scientific workflows are among the most complex workloads and can include extremely complex interconnection of tasks to achieve a single scientific goal. These workflows and the computing resources have been traditionally optimized for performance and throughput.

The general assumption is that a computation with more resources will perform better. This assumption takes into consideration the amount of resources and the performance of the computation. With ever growing concerns for global energy shortages, it is necessary to understand the energy implications of such high performance computations. There are many tools available to improve the performance of compute clusters and workflows but there is a lack of generally available tools for considering and improving the energy consumption of scientific workflows [6].

The aim of this paper is to motivate the development of an adaptive scheduler that takes 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 and possibly the computation level and make changes to the workflow or the cluster to better accommodate the execution of the workflow. A specific set of generic and workflow-specific requirements and policies are also identified. The scheduler makes use of the MAPE model to adapt the configuration of the workflow or cluster to maximize the performance and/or reduce energy consumption.

The key contributions of this paper are as follows: (i) a survey of the current state of energy aware and adaptive workflow schedulers; (ii) identification of run-time adaptations that can be captured using the MAPE model; (iii) a conceptual architecture for an energy-aware adaptive scheduler for scientific workflows and (iv) a proof of concept implementation and evaluation of the adaptive scheduler.

The remainder of this paper is structured as follows: Section II provides literature on scientific workflows, energy-aware workflow schedulers, and autonomic computing. MAPE based possible adaptations that can be done on scientific workflows are presented in Section III. A conceptual architecture for developing an energy-aware adaptive scheduler is provided in Section IV. A proof of concept adaptive scheduler has been developed and evaluated in Section V. Finally, Section VI provides conclusions and future work.

II. Motivation

System management and optimisations of tasks have traditionally been done manually or by complex applications [7]. With ever increasing complexity of tasks and systems in computation, a number of models have been proposed to automate this process. These models have managed to achieve improved performance by adapting to their environment at every level of computation [8]. These models can be used in scientific computation to achieve similar results while focusing on a particular aspect of computation. Based on the current world's increasing energy usage concerns, these models can be used to monitor the energy consumption of the computation and develop energy efficient methods for computing [9], [10]. The remainder of this section provides a background on (i) Scientific Workflows, (ii) Energy-Aware Scheduling, and iii) Autonomic Computing.

A. Scientific Workflow

Scientific computations have grown in complexity and size in recent years [11]. Workflows are a common method to execute and handle complex computations. Workflows comprises of discrete step by step processes for executing the computation. A large computation is broken down into a large number of individual jobs that are executed in batches and parallely. These individual jobs have unique characteristics such as input data, resource requirement, dependencies, etc.

Montage [12], [13] is a software toolkit used in Astrophotography to combine Flexible Image Transport System format (FITS) images of the sky into composite images called *mosaics*. A montage is a series of separate images that are combined to obtain a continuous sequence. Astrophotography makes use of this concept to create high definition mosaics of different astronomical objects such as stars, planets, etc. Highly detailed images of small portions of the sky are obtained and combined to create mosaics of the sky.

The Montage Workflow preserves the calibration and positional fidelity of the original input images [9]. Montage Workflow comprises of different tasks to analyze, combine, verify, export, etc. that develops the relevant mosaic based on the input parameters. Montage workflow comprises on 8 levels of jobs that are dependent on each other. Each level corresponds to multiple instances of a single job that compute different data and execute parallely. The 8 jobs in the Montage Workflow are mProject, mDiffFit, mConcatFit, mBgModel, mBackground, mImgtbl, mAdd and mViewer. Each job is dependent on the previous jobs for their input. Montage has been classified as an input/output-bound workflow [14] compared to other scientific workflows. mProject is the most compute intensive job in the Montage Workflow. Depending on the area of sky to be created, there are two different input variables that can be controlled by the user – the area (*degrees*) of the sky and the colour channels from which the mosaic should be generated.

Many different workflows have been generated for specific computations required by researchers. Workflows are increasingly being used in scientific computation due to its scalable and consistent nature. Multiple jobs in each level of the workflow can be executed in parallel. These parallel jobs scale according to the size of data or the requirements of the user and can span thousands to millions of jobs which can take a long time to execute. Due to the inherent parallelism of the jobs, researchers have developed different techniques to achieve higher performance from the workflow execution. Global energy usage has been on ever rise due to the increasing computation and it is important to focus on energy efficient computing rather than increasing the throughput of a computation [6], [15]–[17].

B. Energy Aware Workflow Schedulers

Parallel computation has always been a go to for huge complex computations. Traditionally, parallel systems have been highly optimized to maximize the performance of the compute resources. With ever increasing need for compute intensive computation and longer execution times, especially in workflow execution, researchers are focusing more on ‘energy efficient computation’ rather than ‘high performance computation’ [9]. Due to this, multiple techniques have been developed to reduce the cost of computation.

[18], [19] developed a scheduling algorithm that tries to minimize the energy consumption of the computation while meeting the time constraints. Similar work was conducted by [20] in which the scheduler made use of polynomial time algorithm to find the optimal resource allocation for a computation in order to minimize the cost of computation.

Edge computing is gaining popularity due to their scalablility and ease of use. Majority of the computation is conducted on remote High Performance Compute Resources. Normally, researchers do not have the necessary knowledge to develop or manage their own computing resources. They depend on remote data centers to execute and manage their computation. Due to their remote nature, edge computing often includes huge data transfers and the inter-dependencies between the tasks act as a overhead. Researchers have been focusing on reducing the energy consumption of edge computations as well. [21] developed a scheduler that reduced data-transfer and inter-dependency of tasks in edge computing. They achieved a reduction of 22.7% in the cost of computation while providing similar performance. [22] developed a scheduling heuristic that reduces the energy consumption of cloud computation conducted on Virtual Machines as well as the systems that are outside the system.An energy-aware task scheduling approach is developed by [23] that dynamically scales the voltage and frequency of edge compute resource in order to reduce the energy consumption of computation.

C. Autonomic Computing

In modern times, computing systems have reached a complexity where the manual effort to keep systems up and functional is getting out of hand [24]. Autonomic computing concerns with automating system management and optimising tasks while adapting to their environment. Autonomic systems have to be manually guided on a high level but are expected to decide the steps that needs to be done in order to keep the systems stable. These systems constantly adapt to the changing environment conditions and adjust their operations to achieve a certain goal.

The adaptation of the system to its environment is managed by an autonomic manager. The manager is effectively a group of control loops that monitor the changes in the system and adapts according to the system goals. MAPE model comprises of multiple stages of control loops where (M) stands for monitoring, (A) for analyzing, (P) for planning, and (E) for execution. A high level description of MAPE model in a compute system is illustrated in Figure 1.

An autonomic system is structured into two components - Managed Resource and Autonomic Manager [7], [24]. The managed resource includes the system or computation that is not adaptive initially but is being managed by the autonomic manager. Autonomic manager makes use of sensors

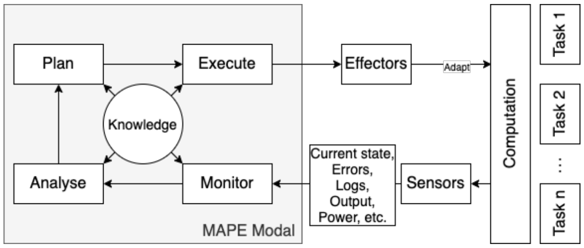

Fig. 1. MAPE Model (Adapted from [2])

and effectors to manage the resource. Based on the data from the sensors, the autonomic manager takes decisions that are implemented using the effectors.

The autonomic manager has four distinct functions which are executed in a continuous loop. These functions are *Monitor*, *analyze*, *Plan* and *Execute*. The model assumes that some *knowledge* of the system is shared and adapted across all the functions.

Monitor uses the sensors from the managed resource and generates data that is consumed by the ***Analysis*** function. These sensors can include any data such as Errors, logs, computation data, system information, etc. All of this data is used in the decision process needed to adapt the resource to its surrounding.

Analysis processes the monitoring data and searches for potential problems, bottlenecks and opportunities for optimisations. The analysis function comes up with all the different changes that can be made to the resource based on its characteristics. These changes don't necessarily have to improve the computation or resource.

Planning takes into consideration the current state of the system, the goal of the system and messages from the ***analysis*** function to decide the appropriate actions to be performed on the resource or computation. The function makes use of predefined policies that are used to achieve the system goal.

Execution makes use of the *planned* changes and the *effectors* to implement the changes. These *effectors* can be in any form such as variables, resources, functions, etc.

Background and literature in the field of scientific workflows, energy-aware computing and autonomic computing has been presented in this section. The current research in the domain of energy aware scheduling are workflow specific and do not take into account the generic optimizations or adaptations that can be made on workflow and cluster level.

III. Energy-Aware Workflow Adaptations as MAPE Based Decision Making

Adaptations to the execution of the workflow or the cluster can be done during run-time with the help of new knowledge obtained from the current behaviour of the computation [25]. This new knowledge can be used to schedule jobs according to a specific goal [26], [27]. The performance of long running scientific workflows stands to benefit from adapting to changes in their environment. Autonomic Computing provides methodologies for managing run-time adaptations in managed systems. A common understood way of achieving this is to use the MAPE model. The following describes the monitoring, adapting, planning and execution for workflow adaptations.

A. Monitoring

Scientific workflows, as shown in Section II, consists of number of inter-dependencies and data transfer between the tasks. These workflows are executed on compute resources. Monitoring different aspects of the workflow execution can help in analysis and planning the adaptive changes required for improvement of workflow. The following is a list of information that can be monitored:

M.1 **Total compute resource** This is a measure of all the computing resources in a compute cluster. This can include both the free and used resources.

M.2 **Available computing resource** This is a measure of all the available computing resources that can be used for executing a task.

M.3 **Progress of the workflow.** This is a measure of how the workflow has progressed. Details of different tasks that have been completed and the tasks that are yet to be completed.

M.4 **Progress of a service.** This is a measure of how the individual task has progressed. It also monitors the input, output and low level computation of the task.

M.5 **Completion of a service.** This monitors the completion rate of tasks. This will focus on individual tasks and help determine the factors affecting their completion rate.

M.6 **Data Usage of a service.** This is a measure of the amount of data or bandwidth that a task is using.

M.7 **Load on an existing computing resource.** This is measure of how a compute resources is being utilized. The amount of resource being used and how much is available for execution of new tasks.

M.8 **Memory usage of an existing computing resource.** This is a measure of the amount of memory being used by a task on the compute resource.

M.9 **Load on the computing network.** This is a measure of the network resources being used.

Monitoring component inside the *Autonomic Manager* collects this data with the help of *sensors* and system log files. Monitoring the available resources and computations can help in determining the appropriate pair of task-resource pair to maximise the throughput of the computing resource. Likewise, monitoring the current state of the computation can help understand the overall factors affecting the performance and can be used to improve other workflow executions.

B. Analysis

Analysis component of the *Autonomic Computing*, as the name suggests, analyzes the data from the *Monitoring* component based on certain rules and specifications and presents the data to the next component in *Autonomic Computing*. A list of possible analysis criteria for the study is provided below:

A.1 **There is a bottleneck.** Bottleneck occurs when a number of different tasks need to be completed in order to progress to the next step in computation. The computation is stuck until all the task-dependencies have been completed. This may be detected by using M.3, M.4, M.5 and M.9 data.

A.2 **There is resource allocation imbalance.** This is where the allocation of resources is not homogeneous. Some tasks may be executed on new resources even though the old resources can execute them. Allocation imbalance may be detectable by M.1, M.2, M.7 and M.8 data.

A.3 **Likely to miss the system goal.** The current progress of the workflow and the progress of each task being executed can help in estimating whether the system goal can be achieved or not. For this, data from M.3 and M.4 can be used to compare with the original schedule.

A.4 **Is there additional free capacity.** This is to measure the amount of resource that is underutilised for extended period of time. This may be detectable by M.2, M.7 and M.8 data

A.5 **There is a new task available.** This determines the next task in the workflow to be executed. This may be done by analysing the M.3 data.

A.6 **There is a new resource available.** This determines if there is any additional resource that is made available for the computing. This may be detected by M.1 and M.2 data.

A.7 **Analyzing the node utilisation.** It may be possible to analyze the individual node utilisation by analysing the data from M.6, M.7, M.8 and M.9 data.

C. Planning

Planning component in the *Autonomic Manager* decides on the necessary adaptations to be conducted in order to achieve the system goal. A set of generic policies and the analysis of the data in Section III-B govern these adaptations. These adaptations may led to the removal of one or more problems (e.g., A.1- A.3) in the computation.

P.1 **To prioritise and remove the bottleneck at the start.** The bottleneck task can be completed on priority basis by completing it's dependencies first. This is in response to the problems detected by A.1 and A.3.

P.2 **To reschedule the task that failed.** Any failed task needs to be analyzed and rescheduled on different resource or configuration based on the reason for its failure. This is in response to problem A.2, A.3 and A.5.

P.3 **To improve the performance by increasing the parallel computing resources.** Based on the type of computation and number of parallel independent jobs, it may be possible to complete the computation faster by increasing the resources to complete the parallel jobs faster. This is in response to A.3, A.4, A.5 and A.6.

P.4 **To make a task complete faster by smartly schedule a task to specific nodes.** Different resources might specialise in different types of executions (e.g. Graphical Processing Unit (GPU) for image processing). Such tasks can be smartly scheduled on unique resources on which they may be able to complete faster and improve the performance of the whole computation. This is in response to A.2, A.5, A.6 and A.7.

P.5 **To assign free resources to the new tasks.** The available resources needs to be smartly allocated to the tasks so as to maximise the utilisation and reduce the overheads. This is in response to problems/ opportunities detected by A.2, A.4 and A.7.

D. Execution

The *Execution* component of the *Autonomic Manager* is responsible for carrying out the plan. This is mainly done by invoking the *Effectors* which can include multiple steps such as suspending the workflow, making changes to the environment variables, upcoming task, cluster resources or hardware components and resuming the workflow.

IV. Energy-aware Adaptive Scheduler

An energy-aware adaptive scheduler for scientific workflows has the potential to substantially reduce the energy consumption of a large amount of computation on clusters. The aims of this paper are to argue for and propose such a scheduler.

A. Requirements for an Energy-Aware Adaptive Scheduler

An energy-aware adaptive scheduler aims to achieve reduction in the energy consumption of the computation by acting and adapting on the current execution data of the computation. To achieve this, the following high level requirements are identified that need to be met:

R1 The proposed adaptive scheduler shall attempt to dynamically reduce the energy consumption of the execution by adapting different policies depending on the current state of the execution;

R2 The proposed adaptive scheduler shall not affect the output of the execution;

R3 The scheduler shall have a set of pre-defined goals that it tries to achieve (percentage of energy consumption reduction, amount of resources to use, etc.);

R4 The proposed adaptive scheduler shall allow the user to specify their own goals;

R5 The proposed adaptive scheduler shall handle any faults in the workflow gracefully and adapt the execution accordingly;

R6 The proposed adaptive scheduler shall make use of a feedback loop to collect the current execution data, analyze it and determine the optimal adaptations in order to achieve the goal set by the user;

R7 The different steps in the decision making of the adaptive scheduler shall be logged for debugging;

R8 The proposed adaptive scheduler shall allow the logged data to be used by different analysis tools.

B. Proposed Energy Aware Adaptive Scheduler

The high level generic design of the adaptive scheduler is presented in Figure 2. The design is guided by the requirements set out in the previous section. The scheduler tracks the current state of the execution with the help of sensors or logs and dynamically optimizes the workflow to achieve the goal set out by the user. FEPAC, a framework developed for collecting and analyzing the execution data [6], is used to create a constant feedback loop between the workflow execution and the adaptations conducted by the scheduler by using the MAPE model (shown by red lines).

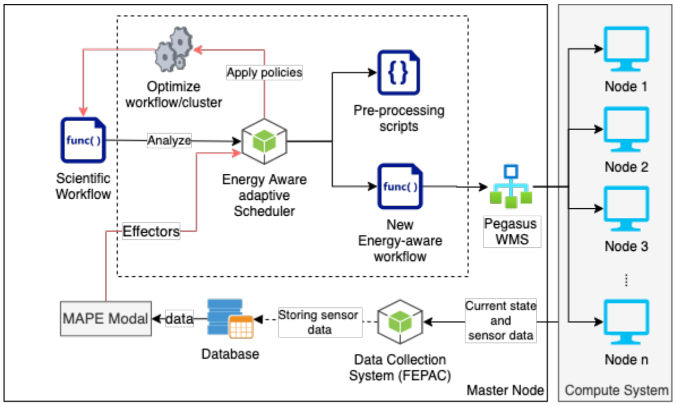

Fig. 2. Energy Aware Adaptive Scheduler

The proposed scheduler, by default, constantly analyzes the current state of the execution for any exploits that can be used to improve the performance or the energy consumption of the workflow. This can be done through number of policies that are pre-defined and known to achieve reduction in energy consumption R1 and R6. The proposed scheduler only affects the scheduling of the jobs and does not change the underlying data or computation of the workflow R2.

A set of predefined goals and thresholds are provided to the adaptive schedule. A user specified configuration file can overwrite these default goals and thresholds. This aligns to the requirement R3 and R4. These goals or thresholds can be a result of budget or compute resources constraints, workflow characteristics, etc. The scheduler does not aim to replace any existing scheduler that is executing the workflow but works in tandem with the scheduler by editing the constraints to the scheduler to achieve the user's goal. The proposed scheduler tests the workflow for any faults before submitting it to the scheduler. The default scheduler can handle any breaks gracefully if occurred R5.

The proposed scheduler logs each step of the MAPE model (monitoring data, analysis output, planning decisions, execution triggers) that is used to perform adaptations. This helps in better understanding which policy was used by the scheduler and how it was implemented in case of debugging R7. Along with this, the scheduler can allow export of the logged data so that it can be analyzed by any third party tools R8.

C. Possible energy-focused scheduling adaptions

The MAPE approach to decomposing adaptations enables a range of adaptations to be supported. The aim of these adaptations is to have an impact on the energy consumption of the executing workflow, ideally leading to a reduction. The following are a selection of possible adaptations which can be supported with a energy-focused adaptive workflow scheduler:

- Change the allocation of workflow jobs to different physical processors to change the energy profile of the job.
- Change the number of jobs allocated to each physical node to change the overall load pattern across nodes.
- Switching on or off physical nodes in order to change the overall available computation capacity or the energy consumption of the cluster.
- Change the allocation proportion to different types of nodes dependent on the performance per Watt for particular jobs of physical nodes.
- React to changes in the usage of nodes which will have an impact on the availability of physical nodes.
- Change the priority of execution of specific workflow jobs to force jobs to be allocated to resources sooner.
- Reschedule workflow jobs if physical cluster nodes are using more than expected energy.

V. Evaluation

A generic design for an energy aware adaptive scheduler is presented in the previous section. To further confirm the functioning of the scheduler, a proof of concept scheduler is developed and evaluated on a scientific workflow in this section.

Single board computers have always been known for their energy efficient nature [6], [9], [28]. For the purposes of this study, a 10 node heterogeneous single board cluster was developed. This was done in order to better understand the effect of different compute resources have on computation. This also help to evaluate the proof of concept scheduler by introducing different compute resources.

The 10 nodes included five Raspberry Pi 4B (Node IDs - 1 to 5) with 2GB RAM, Broadcom BCM2711 Quad core ARM Cortex-A72 processor and five Raspberry Pi 3B+ (Node IDs - 6 to 10) with 1GB RAM, Broadcom BCM2837 Quad code ARM Cortex-A53 processor. The nodes are connected to each other and powered by 2 Netgear GS110TP switches and Power over Ethernet (PoE), respectively. This enabled easy monitoring of their energy usage through the inbuilt sensors of the switches.

The 10 node cluster was managed by a x86 Linux based Intel NUC (8 core with Linux Mint OS (https://linuxmint.com/, accessed on 7 October 2022) that acted as a master node.

The Pegasus Workflow Management System (WMS) was used to manage the submission, management and execution of the workflow on the cluster [29]. Pegasus makes use of HTCondor [30], a batch job scheduler and resource management system, to submit and execute jobs on the cluster. The 1.5 degrees variant of the Montage Workflow (see Section II) was executed on the cluster of 10 nodes. As the Raspberry Pi's (RPis) have quad code processors, HtCondor considers each core as an independent execution resource/ thread. Considering that the cluster is made up of 10 Rpis, there are total of 40 threads that can execute jobs in parallel.

The standard execution of the workflow (seel below) is compared with the execution of the adaptive scheduler and the performance and energy consumption of the computation is collected and analyzed. Two major metrics were used to compare the standard and adaptive scheduler's execution – amount of resources being used (illustrated by Figures 3, 5 and 7), and total number of jobs being executed on the cluster (illustrated by Figures 4, 6 and 8). The jobs that are being executed on the cluster nodes are being considered in the analysis of the data. Master node specific jobs such as file/folder creation or file transfers are not considered.

A. Normal/ Standard execution

Standard execution of the Montage 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 as a baseline for any future comparisons between the different policies of the adaptive scheduler.

The execution data of the experiment is shown below. Figure 3 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 are denoted by different colored dots. The analysis only considers the jobs that are executed on the cluster nodes.

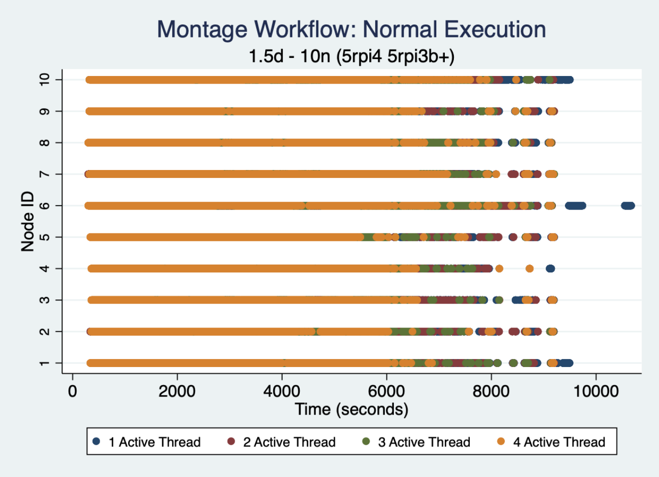

Fig. 3. Number of Active Threads per node for a Montage 1.5 degree workflow for standard execution.

Figure 4 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.

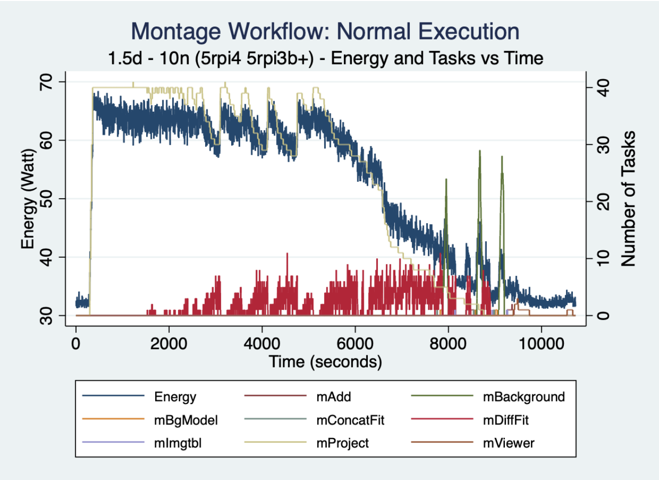

Fig. 4. Number of tasks and Energy Consumption over workflow execution time for standard workflow execution.

The standard execution of the Montage workflow on the heterogeneous cluster took 10,730 seconds (approximately 2 h and 59 min). For majority of the execution, the cluster resources were being fully utilized (denoted by orange color in Figure 3). As seen from Figure 4, the cluster uses maximum energy during the execution of mProject jobs. This is expected as the mProject job is highly compute intensive and maximizes the resource computing power during execution. As the mProject jobs are completed and other jobs start to execute, a drop in the energy consumption of the cluster is observed that shows that the other jobs are comparatively less compute intensive. During the total execution of the workflow, the computation consumed around 154.07 Watt-hr of energy.

B. Adaptive Scheduling - Reducing Energy Consumption - Adapting after 1^{st} set of jobs are completed

A proof of concept adaptive scheduler based on the design in Section IV was developed and the Montage workflow is executed using it. The Adaptive scheduler is configured to achieve reduction in energy consumption of the computation. The scheduler waits for the first set of jobs from different nodes to complete and schedules the remaining jobs according to the analysis of the data of the executed jobs.

The execution data of the experiment is shown below. Figure 5 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 are denoted by different colored dots. The analysis only considers the jobs that are executed on the cluster nodes.

Figure 6 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 jobs being executed on the cluster. Different jobs are denoted by different colors.

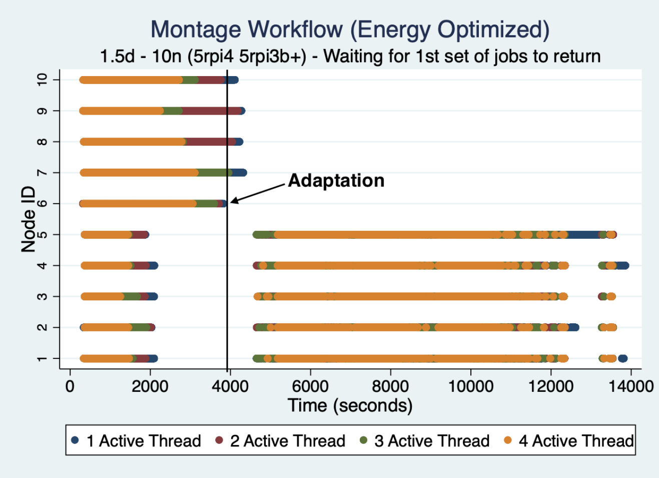

Fig. 5. Number of Active Threads per node for Adaptive Scheduler execution of a Montage 1.5 degree workflow on a 10 node cluster - Waiting for completion of first job on different nodes.

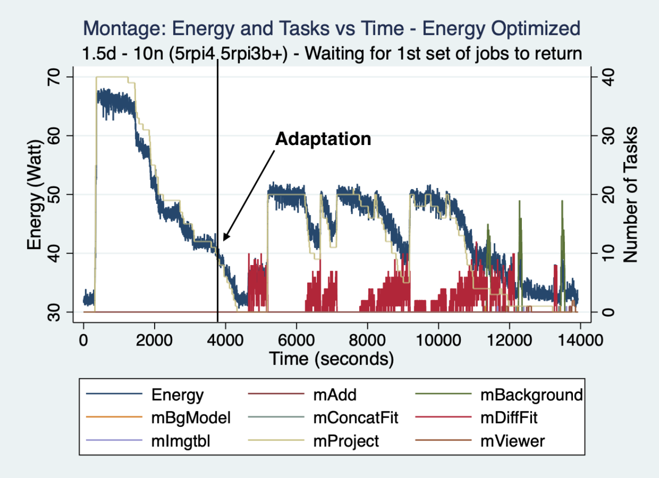

Fig. 6. Number of tasks and Energy Consumption over workflow execution time for Adaptive Scheduler execution of a Montage 1.5 degree workflow on a 10 node cluster - Waiting for completion of first job on different nodes.

As seen in Figure 5, the scheduler refrains from scheduling any new job on the cluster until the first set of jobs are completed. This is the policy that was set at the beginning of the experiment and this helps the scheduler understand the execution power of all nodes and take decisions accordingly. The total execution time of the workflow was 13,636 seconds (approximately 3 h and 47 min). The time does not include the time taken by the scheduler to analyze, adapt the workflow and reschedule the jobs.

The Rpi 4B finishes their first job faster (around 26 min and 20 s) than Rpi3B+ that took around 1 h to complete. The scheduler receives the execution data around that time and reschedules the jobs according to the policies set out in Section IV. In this particular experiment, the scheduler decided that the Rpi 4B, in terms of energy consumption per job, was better at executing jobs than the Rpi 3B+. Thus, all the rescheduled jobs were only executed on Rpi 4B and the Rpi 3B+ were powered off. This lead to the total energy consumption of the 129.66 Watt-hr for the computation. The energy consumption of idle Rpi4B after they complete their jobs are also considered as they have not been powered off during that time.

C. Adaptive Scheduling - Reducing Energy Consumption - Normal execution until the 1st set of jobs are completed

In the previous experiment, it can be seen that the Rpi 4B were idle during the execution of the jobs on Rpi 3B+. To further investigate the energy consumption of the workflow when the nodes are being fully utilized, the following experiment was conducted.

In this experiment, the adaptive scheduler is configured to achieve reduction in energy consumption of the workflow by analyzing the first job being completed on different configuration nodes. But until the first job is completed on nodes that are different, the execution of the workflow is progressed in the standard way. Once the first job is completed on the different node, the scheduler will analyze, adapt and reschedule all the non executing jobs in the workflow. This will make sure that the new jobs are scheduled and executed according to the policies set out in Section IV.

The execution data of the experiment is shown below. Figure 7 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 are denoted by different colored dots. The analysis only considers the job that are executed on the cluster nodes.

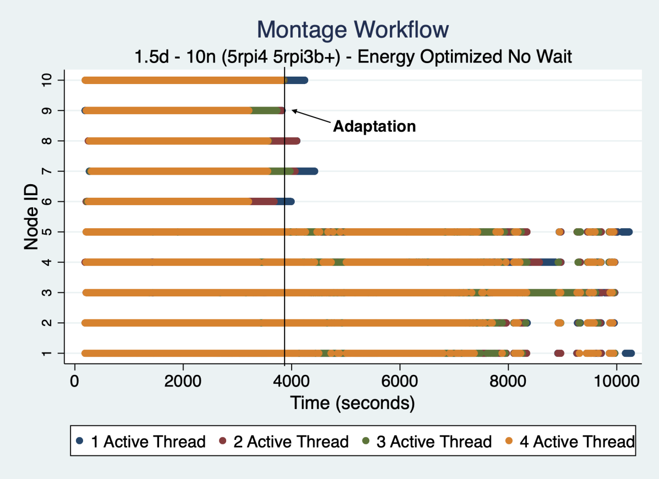

Fig. 7. Number of Active Threads per node for Adaptive Scheduler execution of a Montage 1.5 degree workflow on a 10 node cluster - Normal scheduling until the completion of first job on different nodes.

Figure 8 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 jobs being executed on the cluster. Different jobs are denoted by different colors.

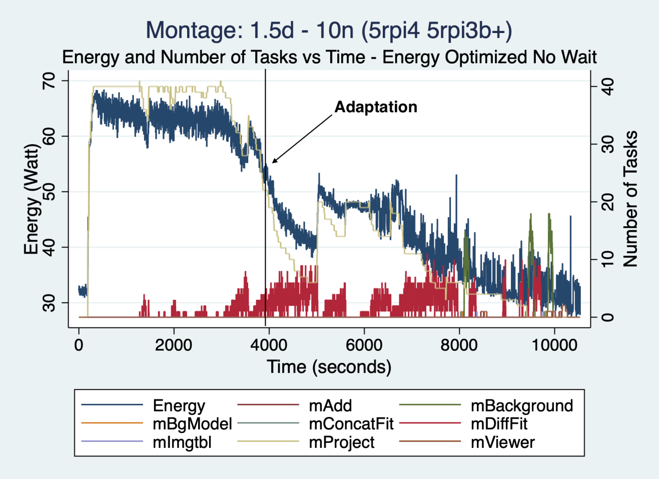

Fig. 8. Number of tasks and Energy Consumption over workflow execution time for Adaptive Scheduler execution of a Montage 1.5 degree workflow on a 10 node cluster - Normal scheduling until the completion of first job on different nodes.

As seen in Figure 7, the scheduler schedules the jobs normally until the first job is completed on the different node. This means that when the first job on Rpi 3B+ completed, the Rpi 4B were executing their second set of jobs. This can be confirmed from the figure that shows that Rpi 4B were always fully utilized. The total execution time of the workflow was 10,531 seconds (approximately 2 h and 56 min). The time does not include the time taken by the scheduler to analyze, adapt the workflow and reschedule the jobs.

The scheduler than makes the necessary adaptations to the remaining non-executing jobs of the workflow according to its analysis. The execution time of each job on Rpi 4B and Rpi 3B+ were the same as the previous experiment. Similar to the previous experiment, the scheduler decided to schedule any new jobs on the Rpi 4B only as it was a better candidate to execute jobs in terms of energy consumption per job. The Rpi 3B+ were considered powered off after the execution of their first set of jobs. The total energy consumption of this experiment was 115.82 Watt-hr.

D. Discussion

The results illustrated in Figure 3 might suggest that the cluster resources are being fully utilized and that this is the the optimal execution solution to the workflow. This is proven wrong by the adaptive scheduler that finds a more optimal scheduling of jobs that leads to reduced energy consumption of the workflow. The standard execution of the workflow took 10,730 seconds with an energy consumption of 154.07 Watt-hr. The first adaptation policy resulted in execution time being 13,636 seconds and consumed 129.66 Watt-hr of energy. The execution time increased around 27% when executed using the adaptive scheduler but a reduction of 15.84% is achieved in energy consumption. This is expected as the scheduler was configured to favoured more energy efficient nodes rather than performance.

Similar results can be found when comparing the standard solution with a different policy of the adaptive scheduler. The second policy execution of the workflow resulted in execution time of 10,531 seconds and an energy consumption of 115.82 Watt-hr. Coincidentally, this policy resulted in an increase in performance as well as decrease in energy consumption of the workflow execution. A decrease of 1.85% was obtained on the execution time and a 24.82% reduction was achieved in the energy consumption. Hence, the third policy resulted in a reduction of *both*, execution time and energy consumption compared to the standard execution.

The performance and energy consumption data obtained from the three experiments conducted align and further solidify the need for an energy aware adaptive scheduler. The evaluation of the proof of concept adaptive scheduler shows that it can help in reducing the energy consumption of the workflow.

VI. Conclusion and Future Work

In this paper, we have argued for the need for adaptive scheduling as an approach for energy-aware scientific workflow execution. It aims to provide mechanisms that reduce the energy consumption of scientific computation without the need for scientists to perform this optimisation themselves. The approach described in this paper works at the scheduler level by retrofitting adaptive behaviour to a cluster scheduler.

The approach is justified through a proof of concept implementation and evaluation. It demonstrated with a real-world case study and cluster setup that using an adaptive energy-aware scheduler can improve the performance and energy consumption of scientific workflow computation. It presented an analysis of energy-aware workflow execution using a cluster scheduler and the reasons for improved energy consumption.

In future work, a more advanced adaptive scheduler will be implemented and evaluated. The focus will be on workflow-specific adaptations that understand the characteristics of particular workflows as they are submitted. More advanced approaches include the ability to schedule multiple different types of workflow simultaneously on a set of shared resources. The aim of this further work is to improve energy-efficient workflow execution in more advanced and realistic scenarios.

References

[1] A. G. Ganek and T. A. Corbi, “The dawning of the autonomic computing era,” *IBM systems Journal*, vol. 42, no. 1, pp. 5–18, 2003.

[2] J. O. Kephart and D. M. Chess, “The vision of autonomic computing,” *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[3] E. Deelman *et al.*, “Pegasus: A framework for mapping complex scientific workflows onto distributed systems,” *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.

[4] 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.

[5] 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.

[6] 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.

[7] K. Lee, R. Sakellariou, N. W. Paton, and A. A. A. Fernandes, "Workflow adaptation as an autonomic computing problem," in *Proceedings of the 2nd Workshop on Workflows in Support of Large-Scale Science*, ser. WORKS '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 29–34.

[8] M. Viroli, A. Bucchiarone, D. Pianini, and J. Beal, "Combining self-organisation and autonomic computing in cass with aggregate-mape," in *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016, pp. 186–191.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] W. Labidi, M. Sarkiss, and M. Kamoun, "Energy-optimal resource scheduling and computation offloading in small cell networks," in *2015 22nd international conference on telecommunications (ICT)*. IEEE, 2015, pp. 313–318.

[17] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, "Energy efficient task allocation and energy scheduling in green energy powered edge computing," *Future Generation Computer Systems*, vol. 95, pp. 89–99, 2019.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Y. Hao, J. Cao, Q. Wang, and J. Du, "Energy-aware scheduling in edge computing with a clustering method," *Future Generation Computer Systems*, vol. 117, pp. 259–272, 2021.

[23] P. Hosseinioun, M. Kheirabadi, S. R. Kamel Tabbakh, and R. Ghaemi, "A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 88–96, 2020.

[24] D. P. Sharma, B. K. Singh, A. T. Gure, and T. Choudhury, "Autonomic computing: models, applications, and brokerage," in *Autonomic Computing in Cloud Resource Management in Industry 4.0*. Springer, 2021, pp. 59–90.

[25] 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.

[26] 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.

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

[28] J. Liu, K. Wang, and F. Chen, "Understanding energy efficiency of databases on single board computers for edge computing," in *2021 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2021, pp. 1–8.

[29] 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.

[30] M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor-a hunter of idle workstations," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 1987.