J Grid Computing (2023) 21:2
https://doi.org/10.1007/s10723-022-09638-7

Developing a Workflow Management System Simulation for Capturing Internal IaaS Behavioural Knowledge

Ali Al-Haboobi · Gabor Kecskemeti

Received: 17 March 2022 / Accepted: 25 November 2022 / Published online: 21 December 2022

Abstract Scientific workflows are becoming increasingly important for complex scientific applications. Conducting real experiments for large-scale workflows is challenging because they are very expensive and time consuming. A simulation is an alternative approach to a real experiment that can help evaluating the performance of workflow management systems (WMS) and optimise workflow management techniques. Although there are several workflow simulators available today, they are often user-oriented and treat the cloud as a black box. Unfortunately, this behaviour prevents the evaluation of the infrastructure level impact of the various decisions made by the WMSs. To address these issues, we have developed a WMS simulator (called DISSECT-CF-WMS) on DISSECT-CF that exposes the internal details of cloud infrastructures. DISSECT-CF-WMS enables better energy awareness by allowing the study of schedulers for physical machines. It also enables dynamic provisioning to meet the resource needs of the workflow application while considering the provisioning delay of a VM in the cloud. We evaluated our simulation extension by running several workflow applications on a given infrastructure. The experimental results show that we can investigate different schedulers for physical machines on different numbers of virtual machines to reduce energy consumption. The experiments also show that DISSECT-CF-WMS is up to 295× faster than WorkflowSim and still provides equivalent results. The experimental results of auto-scaling show that it can optimise makespan, energy consumption and VM utilisation in contrast to static VM provisioning.

Keywords Scientific workflow · Workflow management systems · Simulation · Distributed computing · Energy-awareness · Infrastructure as a service

1 Introduction

Scientific workflows are an increasingly important area in the study of complex distributed applications. Workflows can be run on distributed platforms such as HPC [23, 38, 43], Grid [3, 8, 37] and Cloud [27, 32]. Montage [18], CyberShake [20] and LIGO [1] are scientific workflow applications used in astronomy, earthquake science, and gravitational physics, respectively. They require workflow management systems (WMS) such as Pegasus [18] and Kepler [2])

A. Al-Haboobi (✉) · G. Kecskemeti
Institute of Information Technology, University of Miskolc, Miskolc, 3515, Hungary
e-mail: al-haboobi@iit.uni-miskolc.hu

G. Kecskemeti
e-mail: kecskemeti@iit.uni-miskolc.hu

A. Al-Haboobi
Faculty of Computer Science and Mathematics, University of Kufa, Najaf 54001, Iraq
e-mail: ali.alhaboobi@uokufa.edu.iq

that allow them to run on any of these distributed computing resources. A WMS manages and handles workflow execution through resource selection, job scheduling, appropriate resource allocation and data management. Such systems help to improve performance metrics such as throughput [21, 30], latency [4, 42], and reliability [16, 26].

Cloud computing has become an important platform for executing workflows because it provides the ability to rent resources on demand and in a simple way. Running workflows on Infrastructure as a Service (IaaS) leads to the challenge of determining the number of virtual machines (VMs) to back the jobs of the workflow. At each stage of the workflow, there are a different number of jobs, all of which may require a different amount of computing resources. Static provisioning (i.e., where a fixed set of VMs are provided for the execution of the workflow from the beginning of its execution to the end) may reduce resource waste and financial expenditure, but it can not improve the performance of the workflow. WMSs must not only manage the available infrastructure, but also decide when and how to allocate the resources needed to execute a workflow and how to use them effectively. This requires dynamic provisioning approaches (such as Amazon Auto Scaling[1]) to dynamically add or remove resources based on the workload of the workflow's stages.

Conducting real-world experiments to improve WMS behaviour is challenging when they have to run large-scale workflows. Especially when a statistically significant number of experimental results are required to inform us about possible WMS improvements. This limits the scope of WMS research and development. Therefore, to substantiate research with real measurements, researchers can run a relatively small number of scenarios. Moreover, it is very expensive to reproduce experimental results in different real-world scenarios due to resource costs. Therefore, researchers often turn to simulations.

Simulation is an emerging area for cloud computing that allows evaluating system performance and improve the behaviour of cloud-based applications. Workflow management systems also use simulation to optimise workflow techniques (e.g., scheduling algorithms). Scientific workflow applications can be evaluated in a simulation environment, resulting in a repeatable and controlled environment. Although the accuracy and validity of simulated results always requires final validation in reality, simulation still offers many advantages, such as: reproducibility of results, cost efficiency and flexibility. Although many workflow simulators [11, 22, 40] are available today, they cannot meet the requirements of workflow management systems. These requirements include information about the creation and placement of the virtual machine, as well as scheduling the state of the physical machine. Other cloud simulators [10, 32, 35] are often user-focused and treat clouds as black boxes. Unfortunately, this behaviour prevents the study of the impact of the various decisions made by WMSs at the infrastructure level. Even if a simulator provides insight into the internal workings of clouds, it focuses on some areas (such as precise CPU or network sharing and energy modelling) and ignores others. They thus limit the use cases for these simulators in cloud-aware WMSs [41]. Many workflow simulators [9, 15, 19, 34] do not take into account the provisioning delay of a VM in the cloud. This can have a significant impact on simulation results, especially with auto-scaling that needs to provision and de-provision VMs while a workflow is running in the infrastructure. In contrast to the above problems, DISSECT-CF [27] captures the internal details of cloud infrastructures, which can be used to develop a more informed WMS simulation. It also provides information about virtual machine creation and placement, as well as schedulers for the physical machine. However, DISSECT-CF alone does not provide workflow support.

To address this gap, in this paper, we present DISSECT-CF-WMS, which is built on DISSECT-CF. It was developed to run scientific workflow simulations and investigate internal IaaS behavioural knowledge. First, DISSECT-CF-WMS enables the evaluation of the impact of three physical machine schedulers of a given infrastructure on energy consumption. In addition, it enables better energy awareness by exposing the choice of physical machine schedulers. Second, it is also able to perform large-scale workflows with good execution simulation performance. Finally, DISSECT-CF-WMS has been integrated with the auto-scaling mechanism of DISSECT-CF to execute scientific workflows allowing WMSs to consider the provisioning delay of a VM in the cloud.

Our extension of DISSECT-CF consists of a parser, an engine, a scheduler and a JobRunner component. The parser reads and parses the workflow definition in

[1] https://aws.amazon.com/autoscaling/

the DAG representation and stores information about the tasks and their dependencies. The engine is used to detect when a task is ready for execution and when a workflow is complete. The scheduler receives tasks and schedules them based on a user customisable scheduling algorithm (e.g., HEFT [39], MinMin [5] and MaxMin [6]) to the appropriate resources. The JobRunner orchestrates the execution of the task and its input/output transfers using the core simulator.

We evaluated our extension by running different workflow applications with the three pre-existing physical machine schedulers of DISSECT-CF and comparing their energy consumption. For our evaluation, we decided to use the well-known workflows: Montage, CyberShake, LIGO and SIPHT. This makes our results comparable with future studies. They have been used in the past for various benchmarks and performance evaluations [25]. We have demonstrated the integration of the auto-scaling mechanisms into a larger-scale Montage workflow. Finally, we compared the experimental results of DISSECT-CF-WMS with those of WorkflowSim [15] in terms of simulation accuracy and performance.

The experimental results show that workflow researchers can investigate different PM schedulers of a given infrastructure with different numbers of VMs to achieve lower energy consumption. DISSECT-CF-WMS has better performance than WorkflowSim when the number of tasks in the workflow is increased. The experiments also show that DISSECT-CF-WMS is up to 295× faster than WorkflowSim and still produces equivalent results to it. The experimental results of the auto-scaling mechanism show that the integration has the potential to optimise makespan, energy consumption and VM utilisation compared to static deployment.

The rest of the paper is organised as follows: Section 2 provides background information and the work that currently exists. The details of the design and implementation of the DISSECT-CF-WMS simulator are given in Section 3. We show the performance evaluation of our approach in Section 4. In Section 5, we conclude the paper and future work.

2 Background Knowledge and Related Works

In this section, the scientific workflow concept and some of its scheduling algorithms are presented first. Then an overview of the architecture and functions of DISSECT-CF is given. The section concludes with a problem statement through related work.

2.1 Background Knowledge

A workflow can be represented as a directed acyclic graph (DAG) consisting of a collection of atomic tasks. As shown in Fig. 1, the vertices of the workflow are a set of tasks $\{T_1, T_2, ..., T_n\}$, while the workflow edges represent data dependencies between these tasks. For example, during the execution of the workflow, the successor task T_4 waits for its predecessor task T_1 to complete its processing and produce its output data. When T_1 finishes, some of its data outputs become input dependencies for T_4. When T_4 is scheduled, its data input dependencies are sent to its target host to enable the successful execution of T_4.

2.1.1 Workflow Scheduling

Workflow scheduling is an increasingly important area for WMS. It plays a critical role in the optimal allocation of resources to all tasks. The problem of scheduling in distributed environments is known to be NP-hard [41]. Therefore, no algorithm can achieve an optimal solution in polynomial time, while some algorithms can give approximate results in polynomial time. There are several known algorithms for scheduling; we will use the following three in the rest of the paper:

MaxMin [6] is a three-stage heuristic algorithm for scheduling. First, it filters all ready tasks (i.e., those for which all input dependencies met). Then, it sorts the filtered tasks in ascending order according to the length of their expected running time. Finally, it schedules the task with the longest expected runtime on the best available resource. Consequently, it favours tasks with long runtime over those with short runtime.

MinMin [5] is a very similar heuristic algorithm to MaxMin, the difference being mainly in the sort order: unlike MaxMin, MinMin sorts tasks in descending order (again, by their expected runtime). The task with the shortest expected runtime is then selected to run again on the best available resource. This heuristic aims to create an optimal local path to reduce the total execution time.

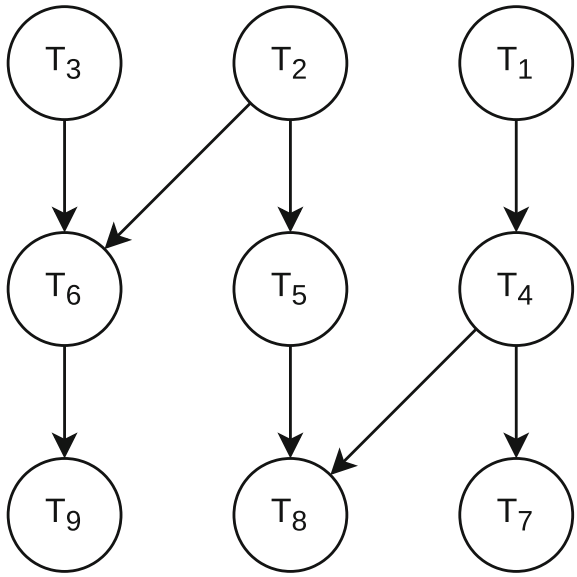


Fig. 1 A sample workflow

Heterogeneous Earliest Finish Time algorithm - HEFT [39]- calculates the average expected execution time of each task on the resources (VMs) and the average communication time of two tasks between all resources. In the first phase, it uses a ranking function to rank the tasks based on the sum of the average execution time and communication time. In the second phase, it assigns a task with the highest ranking value (highest priority) to a resource that would result in minimum execution time.

2.1.2 Infrastructure Simulation

A simulation-based approach is of great interest in the field of scientific workflow research. DIScrete event-based Energy Consumption simulaTor for Clouds and Federations (DISSECT-CF) has been successfully used to simulate the internals of cloud infrastructures. Figure 2 shows the architecture of the currently available[2] 0.9.6 version. The figure groups the main components into subsystems, indicated by dashed lines. Each subsystem is implemented as independently as possible from the others. To perform such simulations, DISSECT-CF has five major subsystems, each responsible for a particular aspect of internal IaaS functionality: (i) event system - for a unified time reference; (ii) unified resource sharing - for solving low-level bottleneck situations; (iii) energy modelling - for analysing the energy utilisation patterns of individual resources (e.g.., network connections, CPUs) or their aggregations; (iv) infrastructure simulation - for modelling PMs, VMs and networked entities; and (v) infrastructure management - to provide infrastructure management

Using these subsystems, simulations can estimate energy consumption and network behaviour, as well as the impact of virtual machine sharing CPU in a variety of scenarios. DISSECT-CF has shown promising performance gains over some popular simulators (e.g. CloudSim, SimGrid). Finally, and most importantly for our workflow simulation goals, DISSECT-CF also provides a simplified network model that allows modelling of data transfers between workflow tasks. As a result, scheduling techniques based on DISSECT-CF can lead to improved workflow execution times.

This type of information is at best only partially available in current commercial and academic cloud (ware) offerings - e.g. Amazon EC2,[3] OpenNebula [31] - but DISSECT-CF enables the analysis of user-side schedulers from new perspectives. Based on the results of this analysis, IaaS providers will be able to provide the most useful information to such schedulers in the future.

To optimally support workflows, cloud systems are often used in the background. To increase cost and energy efficiency, workflow systems could build on auto-scaling mechanisms that are integrated into the clouds. DISSECT-CF's ecosystem also offers several auto-scaling mechanisms that aim to meet the requirements of the application running on the infrastructure. In our case, the application would be either a single workflow instance or all workflow instances managed by a particular WMS. Since the simulator's auto scaling mechanisms are essential for the modern simulation of workflow systems, in the next paragraph we give an overview of the existing approaches provided by the DISSECT-CF-examples project.[4]

The existing auto-scaling mechanisms consider the possible changes to the virtual infrastructure hosted in the cloud every two simulated minutes. Here we discuss the way the changes are decided. The system

[2] https://github.com/kecskemeti/dissect-cf

[3] https://aws.amazon.com/ec2

[4] https://github.com/kecskemeti/dissect-cf-examples

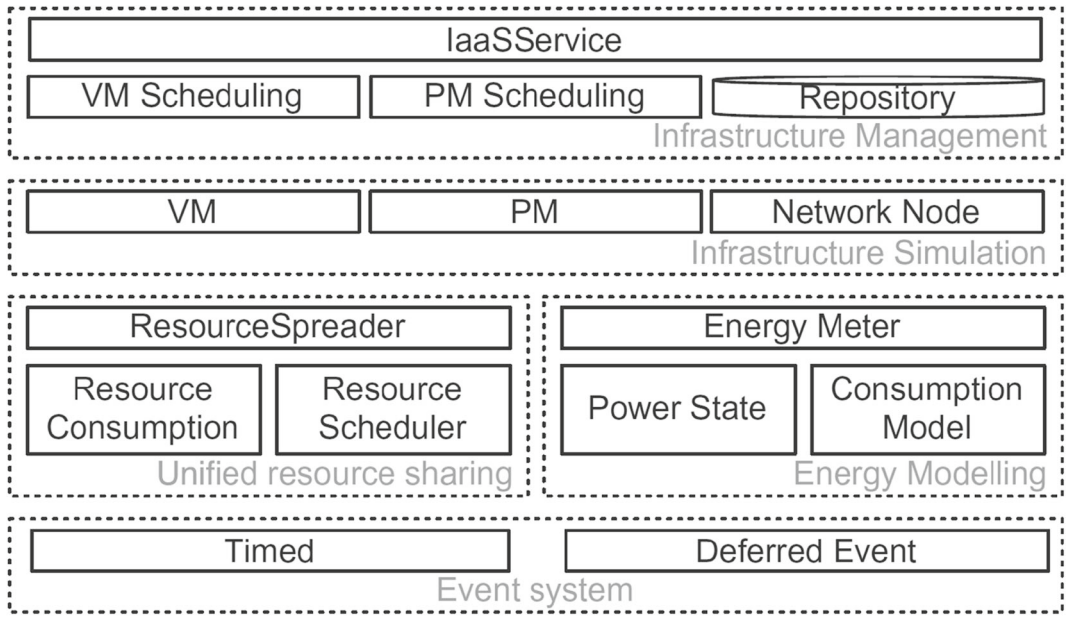


Fig. 2 Architectural view of DISSECT-CF [27]

automatically collects usage metrics (e.g., CPU usage percentage) for all VMs in the virtual infrastructure. There are three different auto-scaling mechanisms that currently use this data. First, the ThresholdBasedVI mechanism destroys underutilised VMs and creates a new one only when all other VMs are heavily utilised. The threshold that determines which VM is underutilised can be set by the user of the mechanism. Secondly, the VMCreationPriorityBasedVI mechanism applies the same thresholds but favours the creation of VMs over their destruction. Finally, the PoolingVI mechanism keeps some unused VMs in the virtual infrastructure. So, unlike the previous approaches that have to queue tasks, the Pooling approach can accept new jobs at any time during the simulation (since it always has some free virtual machines to which it can direct new jobs). As soon as no more tasks need to be executed by the WMS, the virtual infrastructure is completely dismantled.

2.2 Related Works

A number of simulators [15, 22, 30, 33, 40] have been developed for modelling the execution of scientific workflows on distributed platforms such as HPC, Grid, and Cloud. Some simulators [28, 34] have integrated with a particular WMS to obtain more advanced simulation.

Although several workflow simulators [11, 22, 40] exist today, they cannot meet the requirements of workflow management systems. These requirements include physical machine state scheduling, virtual machine creation details, and virtual machine placement. This behaviour does not allow analysing the impact of the various decisions made by workflow management systems at the infrastructure level. In addition, many workflow simulators [9, 15, 19, 34] do not take into account the provisioning delay of a VM in the cloud. This can have a significant impact on simulation results. This is especially true for auto-scaling, which requires VMs to be provisioned and deprovisioned while a workflow is running in the infrastructure.

Workflow integration has been demonstrated for DISSECT-CF by GroudSim and ASKALON [28]. It was shown to be able to improve GroudSim's network model and cloud infrastructure simulation accuracy. This was achieved by introducing internal IaaS behavioural knowledge into GroudSim using DISSECT-CF. The integration allowed ASKALON WMS to interact with the simulated cloud in the same way as with real systems. An evaluation using a 3000-core simulated cloud showed the potential to improve ASKALON behaviour in networking, energy metering, VM instantiation, and CPU partitioning accuracy. On the other hand, this integration caused significant

additional work due to the required joint coordination between the two simulators and ASKALON. In contrast, our newly developed DISSECT-CF-WMS simulator extension relies directly on DISSECT-CF without incurring any coordination overhead. This new direct extension approach also enables the use of previously unavailable auto-scaling mechanisms that can be integrated into the execution of workflow applications on simulation infrastructures.

In [15], the authors presented the WorkflowSim simulator as an extension of the CloudSim simulator. It is designed to run scientific workflows and investigate scheduling and clustering techniques. It includes a task/job fault generator and monitor. It adds queuing/clustering delays to the workflow simulation to more accurately estimate the total execution time of the workflow. WorkflowSim does not capture all the relevant details of the system and its execution [13]. In comparison, we developed a WMS simulator on DISSECT-CF that captures the internal details of the cloud infrastructure and enables the evaluation of WMS execution on three PM schedulers. In addition, DISSECT-CF-WMS has better performance than WorkflowSim when the number of tasks in the workflow is increased. Finally, WorkflowSim does not support an auto-scaling technique or a delay in deploying VMs to the cloud, while DISSECT-CF-WMS does.

In [13], the authors introduced the WRENCH simulator, which builds on SimGrid [12], a versatile, accurate, and scalable simulator. WRENCH was designed to be an accurate, scalable, and easy-to-develop simulation software. It implemented the Pegasus production WMS as a case study. When compared to WorkflowSim, it was found to be slower by a factor of $\sim$1.81 for 10,000 task workflows. This was considered acceptable since WorkflowSim's simulation results were found to be inaccurate. However, we show that our simulator approach is significantly faster than WorkflowSim. Transitively, based on the measurements of [13] measurements, we can conclude that our approach would also be faster than WRENCH.

NetworkCloudSim [19] is a CloudSim extension mainly used for simulating scheduling mechanisms. It does not support dynamic auto-scaling and provisioning delay of a VM in the cloud. In contrast, DISSECT-CF-WMS has dynamic auto-scaling that takes into account the start-up time of a virtual machine in the cloud.

ElasticSim [9] is a toolkit based on CloudSim for simulating workflows with support for auto-scaling techniques. It does not take into account the start-up time of a virtual machine in the cloud, which could have a major impact on simulation results. DISSECT-CF-WMS, on the other hand, has auto-scaling mechanisms that take into account the time needed to provision a VM in the cloud.

To overcome the above limitations, we developed DISSECT-CF-WMS as an extension of DISSECT-CF for analysing internal IaaS behavioural knowledge. This extension enables the evaluation of three physical machine schedulers of a given infrastructure through fine-grained modelling of energy consumption. Furthermore, it is also able to perform large-scale workflows with good execution simulation performance. Finally, it provides an auto-scaling mechanism to dynamically provision and de-provision resources when running workflows, taking into account the provisioning delay of a VM in the cloud.

3 The DISSECT-CF Workflow Management System

This section first explains the design and implementation of DISSECT-CF-WMS. Then the section concludes with the DISSECT-CF-WMS functions for scientific workflows.

3.1 Workflow Management System Simulation

We implemented our WMS simulation approach on DISSECT-CF, a simulator that focuses on internal infrastructure. We chose DISSECT-CF because of its compact API: (i) enables easy extensibility, (ii) supports IaaS energy consumption evaluation, and (iii) enables quick evaluation of different scenarios for IaaS scheduling and internal behaviour. The APIs of DISSECT-CF support modelling of cloud computing, network resources, job executions and file transfers. DISSECT-CF allows the definition of many the types and quantities of physical machines, energy consumption properties and custom VM and physical machine schedulers. In addition, DISSECT-CF provides a virtual machine abstraction that includes migration and consolidation features. DISSECT-CF therefore provides all the basic abstractions required to implement

classes of cloud resources relevant to the execution of scientific workflows.

DISSECT-CF-WMS handles all interactions related to the execution of workflows with DISSECT-CF, e.g., transferring data, executing jobs and completing notifications. The DISSECT-CF-WMS API provides a higher level simulation focused on WMS research. This API provides several relevant higher-level interactions with the DISSECT-CF simulator:

- To characterise the datacentre configurations for the simulated workflows, details of networks, hosts and data centre-level scheduling (e.g. VM placement policies and PM schedulers) must be provided.
- To enable parsing of workflow descriptions. This allows loading and handling of task details and dependencies.
- To provide a custom workflow scheduling algorithm. Researchers can develop new approaches for mapping tasks to the virtual infrastructure supporting the workflow.
- To specify and set up the auto-scaling mechanism that manages the simulated virtual infrastructure hosting and running the workflow.
- To select the time to start the workflow. This helps to identify the transient behaviour of the workflow.
- To instrument the simulation for future analysis. For example, it is possible to configure the collection of details such as the total execution time of a workflow, energy consumption, resource utilisation and information about custom VM and physical machine schedulers.

The shaded part of Fig. 3 shows the main components of the DISSECT-CF-WMS architecture. The figure also shows the main connections between the existing components of the simulator and our new WMS extensions. The figure shows how virtual infrastructures are used by the scheduler to send workflow jobs to. While the figure also shows that the virtual infrastructures are modelled on pre-configured clouds, our additions and their detailed connections with DISSECT-CF are explained in the following subsections. The first three components (parser, engine and scheduler) are built in such a way that they are decoupled from the others as much as possible. Therefore, simulation developers can integrate them into other simulators.

The right side of Fig. 3 shows the relevant components of DISSECT-CF, which manage all models for computation, storage, network and data location. The architectural novelty of our simulation extension is its use of the VirtualInfrastructure class (instead of directly interacting with lower level components), which enables static and dynamic provisioning of VMs. Dynamic provisioning allows down- or up-scaling of VMs while the workflow is running. Virtual infrastructures can instantiate and terminate a specific type of VM when a user's resource needs are more dynamic and sometimes unpredictable. The VirtualInfrastructure class provides interfaces to all relevant internal components to create clouds, storage and virtual machines. The Timed class provides time-related notifications and enables control of the entire simulation. The IaaSService class represents a single IaaS cloud that can host the virtual machines for our virtual infrastructure and workflows. The tasks of the IaaSService are to maintain and manage the physical machines and schedule VM requests among PMs. Through the IaaSService class, DISSECT-CF-WMS supports the cloud as one of the most common execution environments, such as a commercial cloud (e.g., Amazon Web Services (AWS)[5]) and private cloud infrastructures (e.g., those managed by OpenStack[6]). The Repository class represents the storage entities in the system and is responsible for modelling data dependency. Data storage is also simulated by such repositories. The VirtualMachine class simulates the behaviour of a virtual machine on a physical machine.

DISSECT-CF has two types of events: time-dependent and state-dependent. First, the time-dependent events are placed in the event queue of the Timed class. The event subsystem of DISSECT-CF is used to maintain time within the simulated system. Second, the state-dependent events are fired by the entities whose states have been observed. We have used state-dependent events in our WMS that allows to DISSECT-CF-WMS to be notified when a task has been completed during the execution of a workflow. The DISSECT-CF-WMS simulator subscribes to all tasks to be notified when a task is completed.

[5] https://aws.amazon.com

[6] https://www.openstack.org/

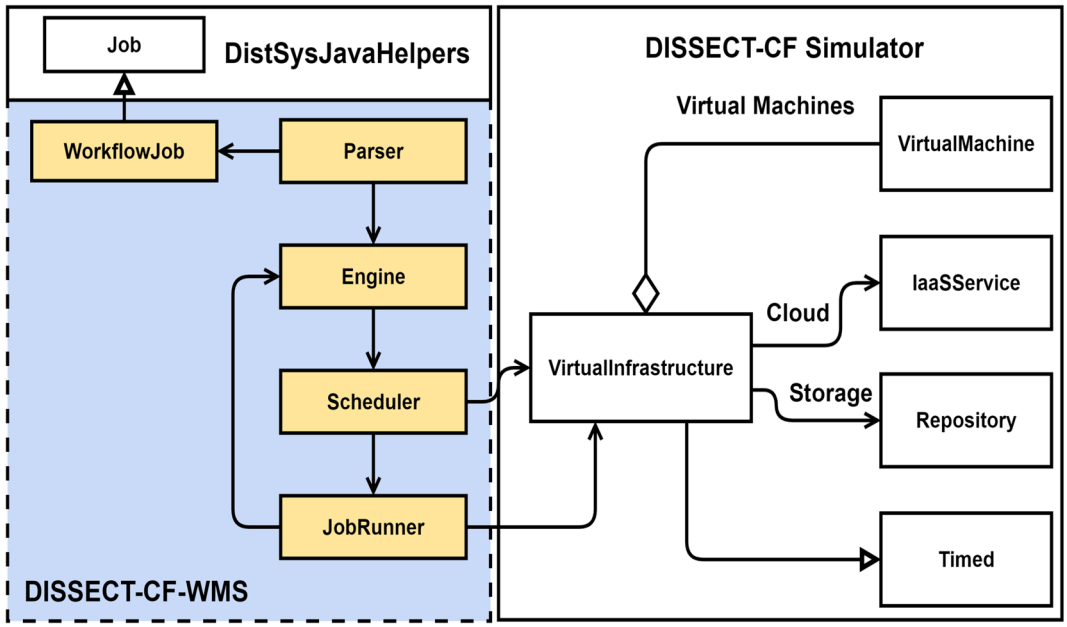


Fig. 3 Class diagram shows the DISSECT-CF-WMS simulator and its connection to DISSECT-CF. The covered area in white colour is supported by the DISSECT-CF simulator and Helper

This allows DISSECT-CF-WMS to provide the execution time of each task within the workflow and the input/output times of the data transfer files of their data dependency.

3.1.1 Parser

The parser component reads the workflow definition from widely used DAX files (Pegasus' workflow description [17]). Parsing creates a list of WorkflowJob instances based on the workflow description files. The WorkflowJob class is an extension of the original Job class of the DistSysJavaHelpers project,[7] which allows the capture of job usage metrics, but lacks the dependency-related information required for workflows. Each instance of WorfklowJob stores the important information needed to process each task, such as runtime, predecessor tasks and data dependencies (input/output files). Some tasks have one or more data files that are not from their predecessor tasks. Therefore, we modelled these task inputs by transferring them from a data staging site to the selected task execution site (VM). Generally, the data staging site is a shared file system at the execution site, such as NFS, or in some cases a file storage service near the execution site, such as Amazon S3. This site is modelled as a central data storage and it is used to stage data in and out for a workflow. Then, the next component will maintain task dependency constraints for managing the scheduling process.

3.1.2 Engine

After the Parser component reads the workflow definition information, the Engine component receives and processes the information by checking the predecessor tasks of each task. The Engine has information about the complete structure of the workflow. It's main task is to determine which tasks are ready for execution. A task can only be ready in two ways: (i) A task without predecessors is always ready. (ii) A task with predecessors is only ready when all its predecessors have finished their execution.

To simplify the engine's task, our engine only checks the first readiness criterion on its own. To ensure that we still process all ready tasks of the second criterion, we ensure that DISSECT-CF notifies the engine of task completion for all previously detected ready tasks. When the engine receives notification of task completion, it updates the task's successor tasks and removes itself from the predecessor list of successors. This allows the successors to be eligi-

[7] https://github.com/kecskemeti/DistSysJavaHelpers

ble for scheduling by the workflow scheduler chosen by the user of the simulator. Our WMS extension also provides different task states. Figure 4 shows the sequence of task states from an unavailable state (not a ready task) to a ready state when all predecessor tasks have been completed. Next, it is either in a running state if it is scheduled on a resource (VM) or in a waiting state if no resource is available.

One of the problems in developing a WMS is how to deal with failures. Computing resources have a small likelihood of failure during the execution of the WMS. As demonstrated via the DCF-Exercises project,[8] DISSECT-CF is able to mimic arbitrary infrastructure failures by using a random failure generator. Our WMS extension builds on this capability by also specifying a failure cause for each failure, that allows triggering a task if its computation fails. This capability is introduced in the Engine component, which monitors the status of a task and takes appropriate action, i.e., a failed task is automatically resubmitted for execution after a timeout. For example, if a task fails, our WMS sends it to the queue and resubmits it to another computing resource. This method is part of DISSECT-CF-WMS and can be used to simulate VM failure probability and error handling capabilities for simulated workflow executions. This can be used to create more robust fault tolerance mechanisms. Finally, the task is in its completed state when it has successfully finished its execution.

3.1.3 Scheduler

The scheduler receives ready tasks in a local queue from the Engine component to schedule them. In this step, we need to select appropriate resources for the ready tasks using the scheduling algorithm specified by the user. This algorithm can be implemented by the user, or it can be one of the pre-implemented solutions, i.e., HEFT [39], MinMin, [5], and MaxMin, [6] (these were introduced in Section 2.1). We support two types of scheduling algorithms: static (e.g., HEFT) and dynamic (e.g., MaxMin, MinMin and DataDependency). Static algorithms start with the assignment of tasks to VMs in the workflow planning stage. Assignment of tasks to VMs occurs before the start of workflow execution. In this case, the Engine component is still responsible for releasing tasks whose predecessor tasks have completed execution. But the Scheduler component assigns a task to its assigned resource beforehand. In contrast, the dynamic algorithms start assigning tasks to VMs during workflow execution. Tasks are assigned to VMs when the tasks are ready and VMs are free during the workflow execution phase. We have developed the DataDependency scheduling algorithm that takes into account data transfers. It selects several ready tasks from a list of task objects stored in the data structure for execution based on the free available resources (VMs). The scheduler component is also responsible for storing information about which VM to execute each task on. This informs the JobRunner component about the location of files with data dependencies (e.g., where the predecessors stored their outputs). This step enables the actual execution of the task on a virtual machine, which is covered in the next section.

3.1.4 JobRunner

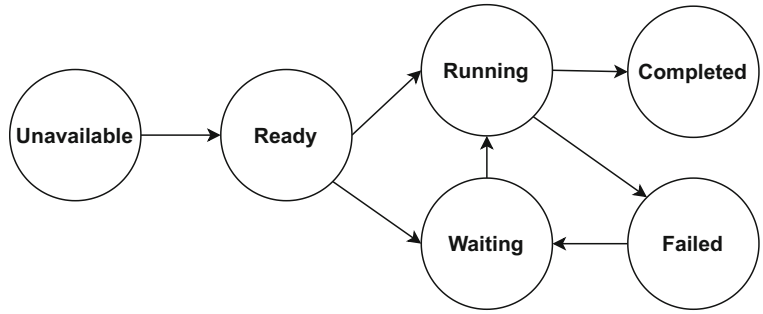
The JobRunner component is responsible for the execution of each task on a previously selected resource (VM). It transfers all files on which a task depends to the execution VM. The transfer is done using the network APIs of DISSECT-CF. If a task is assigned to the same VM that executed a predecessor task, the corresponding dependency transfer does not take place. After notification of the completion of the dependency transfer, the execution of the task on the VM begins. The task-level resource sharing and execution model is provided by DISSECT-CF. This allows DISSECT-CF-WMS to obtain accurate and reliable information about task completion. If necessary, JobRunner transfers the output files from an execution site to the central storage site (staging data out) after a task is completed. After all these activities are completed, the JobRunner notifies the Engine component that the task is complete. This step enables the engine to schedule successor tasks.

3.1.5 Dynamic Behaviour

Figure 5 shows the basic interaction required in our extension to execute a single workflow task. DISSECT-CF-WMS simulates the exchange of a large

[8] https://github.com/kecskemeti/dcf-exercises

Fig. 4 A task state diagram



number of messages between its components about the state of the task. When the process is complete, the Engine component receives the details of the workflow information in a data structure. It then forwards the ready tasks to the scheduling process. Before scheduling takes place, the Scheduler retrieves information about the available resources (VMs) from the VirtualInfrastructure component of DISSECT-CF. Based on the dynamic information about resource availability, a ready task is assigned to a VM using the selected workflow scheduler. Then the JobRunner component manages the transfer of files with data dependencies to an execution site (VM) to start the execution of a task. Finally, the JobRunner component sends a task completion notification to acknowledge the engine component. This allows the successor tasks of the completed task to update their precedence conditions and be ready for scheduling. Therefore, the scheduling process continues till all tasks are scheduled. The Engine component is also responsible for completing the execution of the workflow. It counts the number of completed tasks. Once all parsed tasks have received a completion notification of execution from JobRunner, the Engine shuts down the other WMS components (i.e., the Scheduler and the Job Runner) associated with the execution of the workflow.

3.2 Auto-Scaling Mechanism

We integrated the DISSECT-CF-WMS simulator with the existing auto-scaling mechanisms of DISSECT-CF.

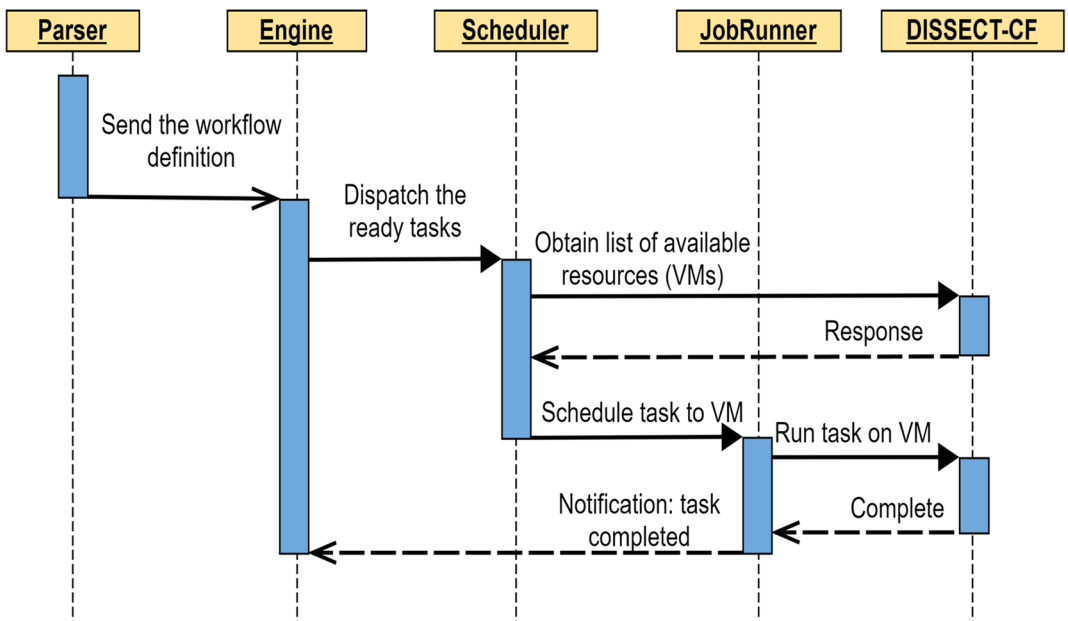


Fig. 5 The interactions between the DISSECT-CF-WMS components and DISSECT-CF for the task's lifecycle till it completed

We have adapted DISSECT-CF-WMS to provide auto-scaling for a workflow execution environment. We have taken into account the delay in provisioning a VM in the cloud, which can have a significant impact on simulation results. After a virtual machine is requested, it is not immediately available for use. The provisioning delay of a VM is the time it takes to be provisioned and booted on a physical host. This enables analysis of dynamic provisioning of resources while running scientific workflows in the cloud to overcome issues of under- or over-utilisation of resources. The auto-scaler behind our WMS extension provides dynamic provisioning and de-provisioning of the number of VM instances based on user-selected criteria. We have integrated our WMS into the virtual infrastructure of an auto-scaler, which is able to automatically scale up or down resources based on the auto-scaling approach to better meet the demands of newly arrived tasks. We modified the JobRunner component to accommodate data transfers. Since the auto-scaled virtual infrastructure creates and destroys VMs at will, the memory of these VMs is volatile and cannot be used for long-term storage of data dependencies during workflow execution. Therefore, our approach places data files in a central data storage for staging data to and from for a workflow. DISSECT-CF provides three basic mechanisms for auto-scaling (we discussed these in detail in Section 2.1). When configuring workflow experiments, the auto-scalers can be selected and their effects on the WMS analysed.

Auto-scaling provides a dynamic and scalable way of scheduling multiple workflows simultaneously with different virtual machine images to facilitate the execution of a number of tasks from a variety of workflow applications. Users can develop novel auto-scaling policies by extending the base VirtualInfrastructure class to override its methods, such as the three mechanism classes (PoolingVI, VMCreationPriorityVI, and ThresholdBasedVI), as shown in Fig. 6. Users can develop an approach to store their intermediate data on the VMs used for execution, but the data on a particular VM should be moved to central storage when a mechanism needs to de-provision that VM. Some users require a dynamic provisioning technique for developing some workflow scheduling algorithms that need this technique. This concept is applicable to algorithms that use either static or dynamic resource provisioning. This technology allows algorithms to dynamically adjust the number and type of virtual machines used to schedule jobs while workflows are running.

DISSECT-CF-WMS can query the CPU utilisation for any period during workflow execution to identify the current VM utilisation pattern. Therefore, this behaviour results in either deprovisioning some unused VMs or provisioning VMs when the current VM utilisation is high, e.g. when the three auto-scaling mechanisms use this feature (VM request, VM termination). More mechanisms could be added to reflect the environment in real life.

4 Evaluation

We demonstrate the capabilities of DISSECT-CF-WMS using the following evaluation experiments. First, we evaluated how the pre-existing three physical machine schedulers influence the energy consumption of various workflows. Second, we compared the simulation of DISSECT-CF-WMS with WorkflowSim in terms of simulation accuracy and performance. Third, we have shown the advantages of using the auto-scaling mechanisms of DISSECT-CF-WMS to

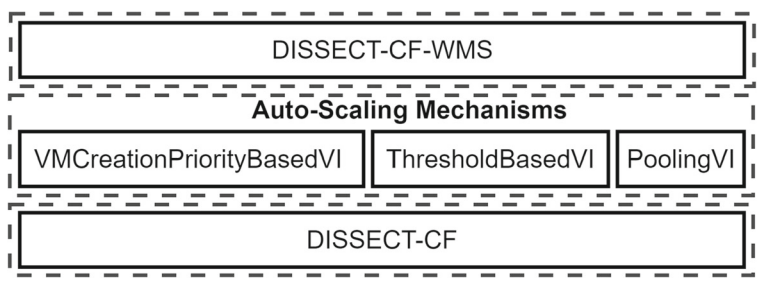


Fig. 6 The overview of the DISSECT-CF-WMS simulator integrated with the auto-scaling mechanisms of the DISSECT-CF simulator

optimise makespan, energy consumption and VM utilisation over static provisioning. Finally, we evaluated the three built-in scheduling algorithms with four popular workflow applications in terms of energy consumption. The simulations were run on a laptop with 12 CPUs of Intel Core i7-8750H CPU @ 2.20GHz, 16GB RAM and 119GB SSD.

All experiments were evaluated with synthetic workflows derived from the Montage (astronomy), CyberShake (earthquake science), LIGO (gravitational physics) and SIPHT (biology) applications, taking into account data transfers. The Montage [18] workflow is an astronomy application used to generate custom mosaics of the sky based on a set of input images. The CyberShake workflow is used to characterise earthquake hazards by generating synthetic seismograms. The Laser Interferometer Gravitational Wave Observatory (LIGO) [7] workflow is used to analyse data from the coalescing of compact binary systems such as binary neutron stars and black holes. The sRNA Identification Protocol using High-throughput Technology (SIPHT) programme [29] uses a workflow to automate the search for sRNA encoding- genes for all bacterial replicons in the National Center for Biotechnology Information (NCBI) database. Figure 7 presents the structure of four workflows.

To simplify the configuration of the simulated cloud, we used the DCCreation class from DISSECT-CF. We configured the simulated infrastructure for our WMS experiments by setting up a homogeneous cloud with 100 physical machines (each configured with 32 CPU cores, 256 GiB of memory and 256GB of storage, and a linear power model ranging from an idle power draw of 296 watts to a maximum power draw of 493 watts) and also configured a central data storage of 36 TB. The machines and central storage were simulated to be connected via a single switch (we set the bandwidth between the machines and the switch to 2 Gbit). All created physical machines are connected via a cloud-level network. The physical machines are potentially controlled by three physical machine schedulers.

To evaluate the energy efficiency, we used the IaaSEnergyMeter class from DISSECT-CF, that allowed us to monitor the energy of the entire IaaS system generated by the DCCreation class. We set up our energy metre to monitor the entire cloud and collect energy-related details in every simulated hour. In addition, we used the HourlyVMMonitor class, which can monitor the utilisation of each VM's CPU in an hourly rate.

We collected the data centre metrics after the first task of the workflow was started. We instrumented

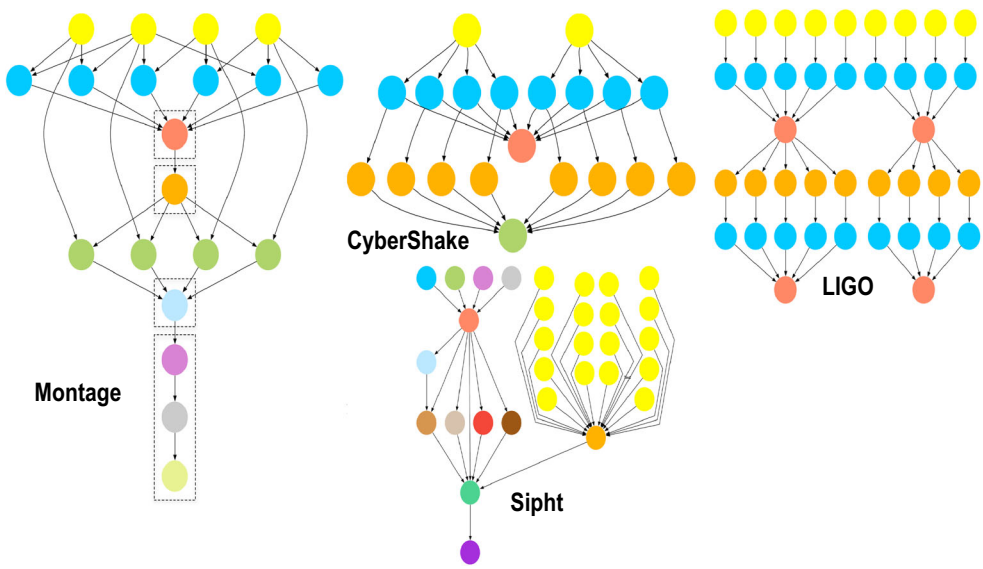


Fig. 7 The structure of the Montage, CyberShake, LIGO and Sipht workflows

the following simulation experiments to capture the following metrics: (*i*) the makespan (total workflow execution time), i.e., the start time of the first task to the completion time of the last task, (*ii*) the average VM utilisation, i.e., the average of the hourly reports for each VM during the complete execution of a workflow, and (*iii*) the total energy consumption of the data centre in kilowatt hours (kWh) as reported by the IaaSEnergyMeter.

4.1 Utilisation of Internal Cloud Infrastructure Details

We configured a virtual infrastructure with a static number of VMs (in a single experiment, we set the number of VMs between 30 and 100; all VMs were homogeneous in terms of the number of CPU cores and memory). We used FirstFitScheduler as VM scheduler and the DataDependency algorithm as task scheduler on VMs. The FirstFitScheduler is a VM scheduler that implements one of the simplest VM schedulers. It places each VM at the first PM that would actually accept it. We ran each static virtual infrastructure on the cloud mentioned above, but we replaced the schedulers for the physical machines with the three offered by the simulator: (*i*) AlwaysOnMachines (AOM), (*ii*) SchedulingDependentMachines (SDM) and (*iii*) MultiPMController (MPMC). First, AlwaysOnMachines ensures that all PMs are controlled so that they always remain on. Second, SchedulingDependentMachines increases or decreases the power of the PM set according to the requirements of the VM scheduler (this scheduler changes the power of the PM set by one PM at a time). Finally, MultiPMController is very similar to SDM, but immediately increases the number of machines needed to run the current infrastructure (i.e. if 4 newly powered-on PMs are needed to host the current demand of VMs, all 4 are powered on immediately). We set a linear model for DISSECT-CF-WMS, which assumes that power consumption depends on the degree of use of CPU, ranging from an idle power consumption of 296 watts to a maximum power consumption of 493 watts. We recorded the power consumption for DISSECT-CF-WMS from the start time of the first task to the completion time of the last task of the workflow.

Figures 8, 9, 10 and 11 show the collected energy consumption for each experiment of DISSECT-CF-WMS when running 1000 tasks each of the Montage, CyberShake, Sipht and LIGO workflows. With a small number of 30 VMs (4 cores) using only slightly less than 4% of the total infrastructure, the MPMC and SDM schedulers have much better energy consumption than the AOM scheduler (i.e., they consume more than 11 *times*, 15 *times*, 7 *times*, and 8 *times* of energy for the same computation of the Montage, CyberShake, Sipht and LIGO workflows, respectively). This pattern repeats (with smaller advantages) for almost all larger VM numbers, except when the VMs use the entire infrastructure. In all cases, AOM's

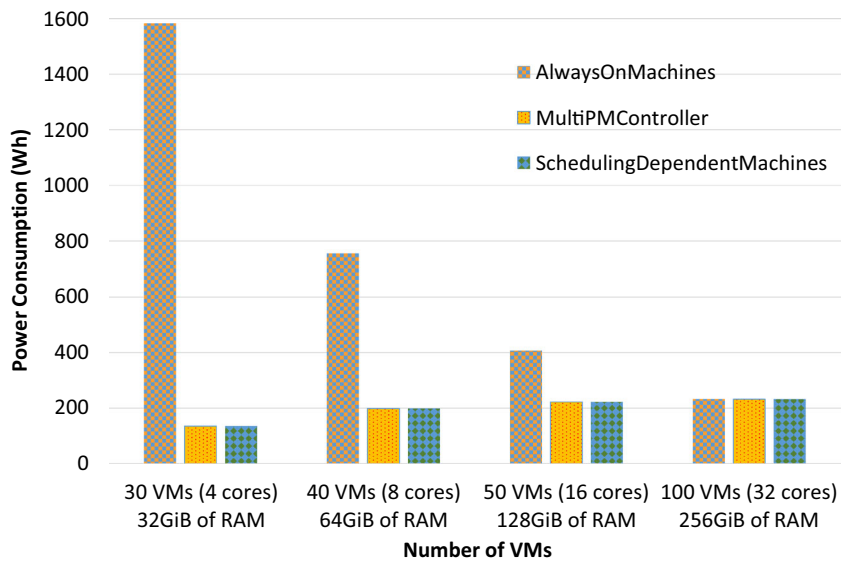


Fig. 8 The total power consumption of PM schedulers for the Montage workflow on DISSECT-CF-WMS with different numbers of VMs

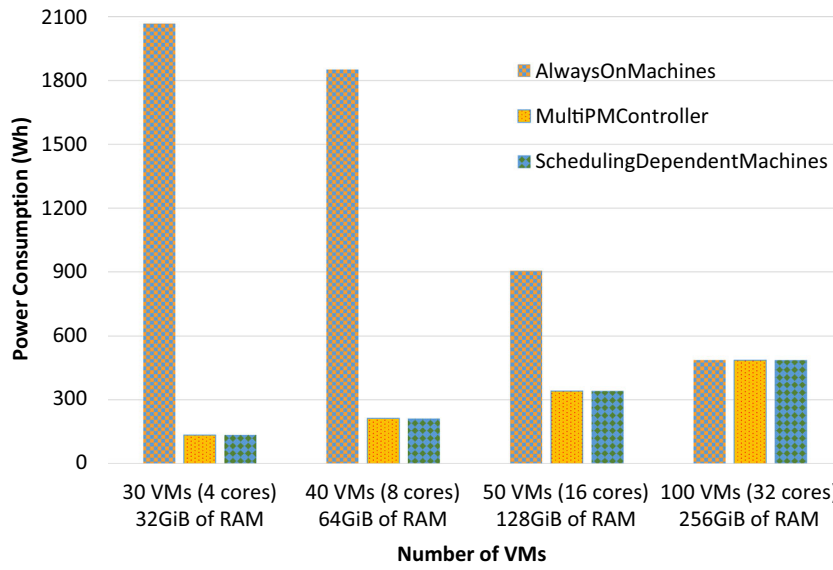


Fig. 9 The total power consumption of PM schedulers for the CyberShake workflow on DISSECT-CF-WMS with different numbers of VMs

strategy of switching on all machines regardless of workload pays off, as it makes all VMs available for workflow at the earliest opportunity. In contrast, the SDM and MPMC schedulers achieve a large reduction in energy consumption, as SDM's strategy of switching on all machines in the data centre one by one (at a time) results in a long overall simulation time due to the provisioning delay, as shown in Fig. 12. The MPMC policy, on the other hand, immediately switches on the number of machines needed for the current operation of the infrastructure. Static VM allocation policies for workflows are not suitable for data centres using a PM scheduler such as SDM.

First, AOM has the same energy consumption patterns for all workflow applications because it never considers switching off machines and thus it results in energy consumption for the entire infrastructure, even for the PMs that do not host VMs. Second, all PM schedulers have the same energy consumption when using the entire infrastructure, as they use all

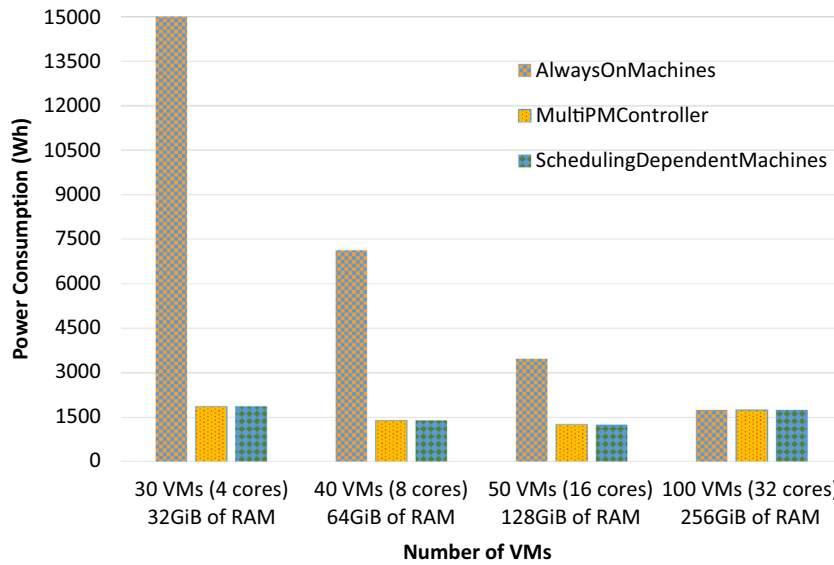


Fig. 10 The total power consumption of PM schedulers for the Sipht workflow on DISSECT-CF-WMS with different numbers of VMs

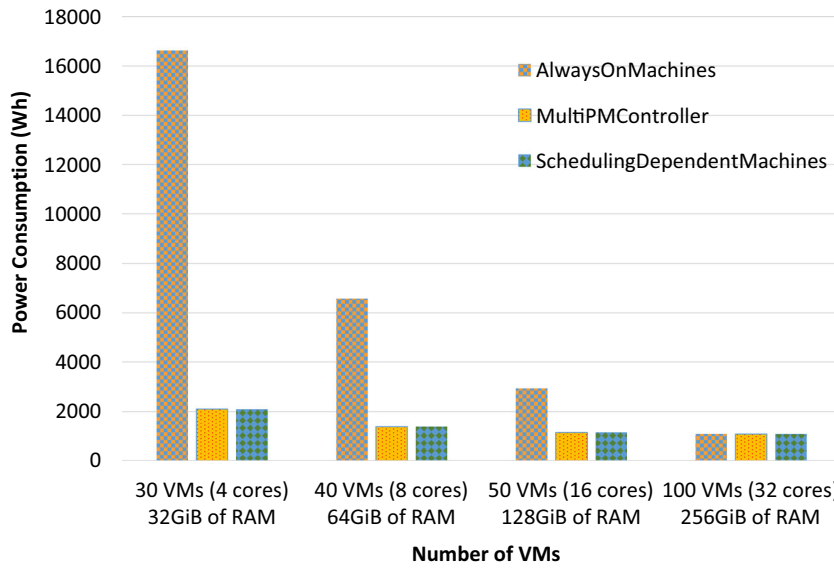


Fig. 11 The total power consumption of PM schedulers for the LIGO workflow on DISSECT-CF-WMS with different numbers of VMs

machines. Moreover, the MPMC and SDM schedulers have similar patterns for the Montage and CyberShake applications. However, the Sipht and LIGO applications have reduced energy consumption by increasing the number of VMs because they have not used all the statically created VMs at all times, except for the Sipht experiment with 100 VMs, which only uses a maximum utilisation of VMs 70% because Sipht has some tasks with significant differences in their running times, so that the time difference can be as much as 19×. This results in idle time in other resources and scheduling gaps between tasks in the workflow, leading to the highest energy consumption. This pattern is repeated in LIGO's experiments, but the time difference can be as high as 3×, resulting in lower energy consumption.

If we compare the experiments from the cloud users' point of view, the results show the advantage of the AOM and MPMC schedulers. As our base WMS waits for all statically allocated VMs to start up, the VMs behind our workflows can start faster thanks to AOM's always-ready physical machines. This reduces

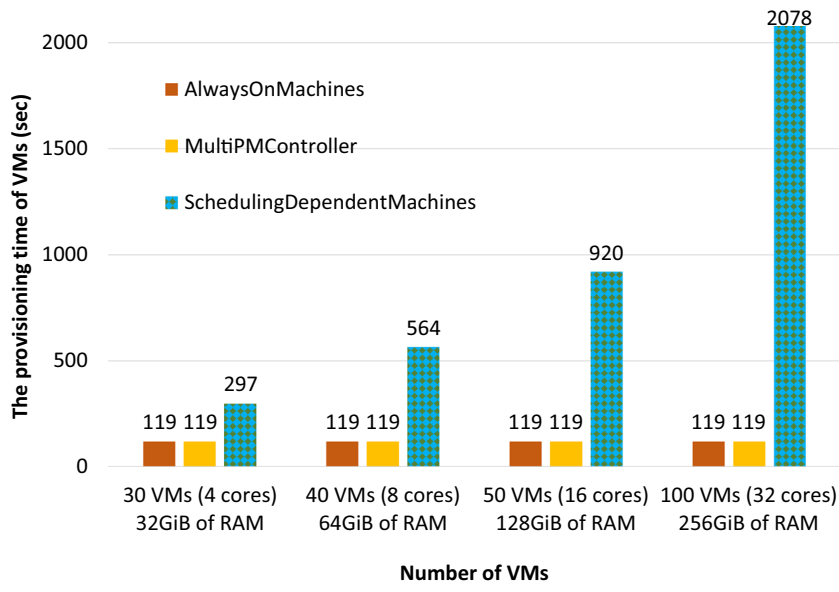


Fig. 12 The provisioning time of VMs for three PM schedulers on DISSECT-CF-WMS with different numbers of VMs

VM provisioning time, as shown in Fig. 12. Note that despite AOM's significant energy penalty, the improvements in provisioning time are equally significant. The weakness of the SDM strategy is also evident in the waiting time. Our WMS has to wait significantly longer for the requested VMs to be ready before it can assign tasks to them. The waiting time difference can be as high as 17× as shown in Fig. 12. The differences are mainly due to the fact that SDM is very slow in starting machines. As a result, the execution of the entire workflow is delayed with many fewer physical machines turned on (but those few are turned on for a significantly longer time, as shown by the provisioning times in Fig. 12). These differences show that for dedicated private cloud infrastructures it is advisable to switch on all PMs required for the workflow. This way we get the results back the fastest and also do not consume too much energy during the runtime of the workflow, like the MPMC scheduler. Thus, DISSECT-CF-WMS is able to offer insight for analysing different workflow execution scenarios and instrument the execution environment to gain insight into the impact of the chosen infrastructure configuration.

4.2 Simulation Times

Now that we have demonstrated the benefits that a WMS simulation extension can provide for the evaluation of WMS behaviour, we move on to the evaluation of the core functions of the WMS. We have compared the performance and accuracy of the simulation results of our system with version 1.1.0 of WorkflowSim, using the same laptop as mentioned above. DISSECT-CF-WMS does not use logging mechanisms, but we printed the execution details as messages. To ensure a fair comparison, WorkflowSim's logging mechanisms were disabled. We made sure to run two experiments, one in each simulator with exactly the same settings. First, we made sure that the simulated data centres had the same characteristics. Again, we used the same cloud mentioned earlier. We requested a static VM configuration that occupied the entire data centre: 100 virtual machines with 32 cores each. For our WMS, we used FirstFitScheduler as the VM scheduler, AOM as the PM scheduler and DataDependency as the scheduling algorithm. For WorkflowSim, we used DATA as the scheduling algorithm, LOCAL as the local file system for storing the data dependency files and the time-shared model as the policy for VMs and jobs. We have now evaluated both simulators with synthetically generated Montage workflows of different sizes (the number of tasks ranged from 1K to 15K).

We compared the total execution times reported by both simulators for all the workflows. We have obtained very similar execution times in both simulators. The difference between the two had a mean absolute percentage error (MAPE) of less than 0.16%. This difference in execution time is due to the fact that our workflow scheduler in DISSECT-CF-WMS assigns tasks to VMs in a slightly different way than the approach taken by WorkflowSim. As a result, the transfer time of the dependent data may be different. Despite the accurate and more detailed simulation (i.e., we provide more insight into the internals of the data centre behind the workflow), DISSECT-CF-WMS delivers the results in significantly less time. Figure 13 illustrates the performance differences between the simulators. We see that our measurements of the real duration of the simulation show that the performance advantage of DISSECT-CF-WMS is between 18 and 295× (i.e., we can get to the same quality results at most 2 orders of magnitude faster). Moreover, WRENCH took 13 minutes to simulate an Montage workflow with 10,000 tasks [14], while DISSECT-CF-WMS took about 5 seconds to simulate the execution of the same workflow.

WorkflowSim builds on CloudSim, which uses a process-based paradigm where each entity in the system has its own thread, resulting in poor scalability as the number of entities in the system grows [36]. DISSECT-CF, however, requires only one simulation thread (instead of one thread per entity). As a result, our WMS outperforms WorkflowSim, as shown in Fig. 13.

4.3 Simulation versus Execution

The previous simulation experiment demonstrated the performance and accuracy of our system's simulation results to WorkflowSim. We compared our simulation result to an existing execution of a real-world Pegasus workfolw (Montage-2.0) on the AWS-m5.xlarge platform to validate the simulation environment [14]. The Montage workflow contains 1240 tasks, and we compared the real execution of five traces to the simulated one. We replicated the identical execution

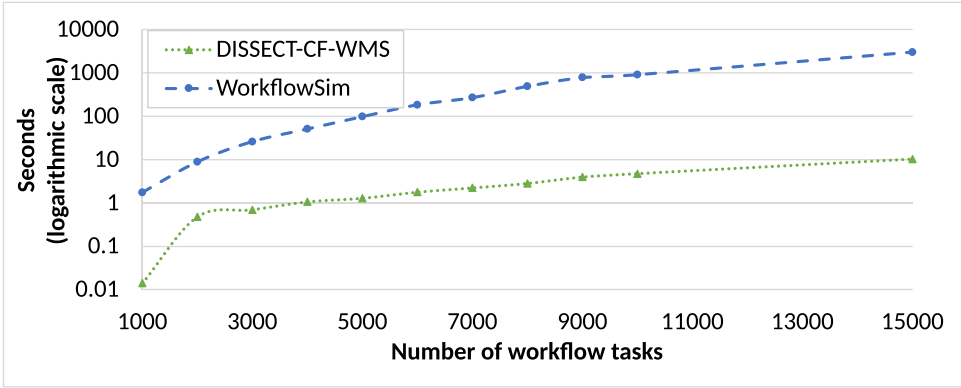


Fig. 13 The simulation time of DISSECT-CF-WMS and WorkflowSim simulators with different numbers of tasks in Montage workflows

environment, which performs similarly to Amazon EC2 m5.xlarge instances. The execution environment includes a submission node that runs Pegasus and DAGMan, as well as four worker nodes (4 cores per node with a shared file system). In these instances, the bandwidth between the data node and the submit node was 0.44 Gbps, while the bandwidth between the submit and worker nodes was 0.74 Gbps and 1.24 Gbps, respectively. Figure 14 depicts Gantt charts of the real execution, whereas Fig. 15 depicts the simulated execution. On the vertical axis, task executions are shown as a line segment on the horizontal time axis, covering the time interval between the task's start and end

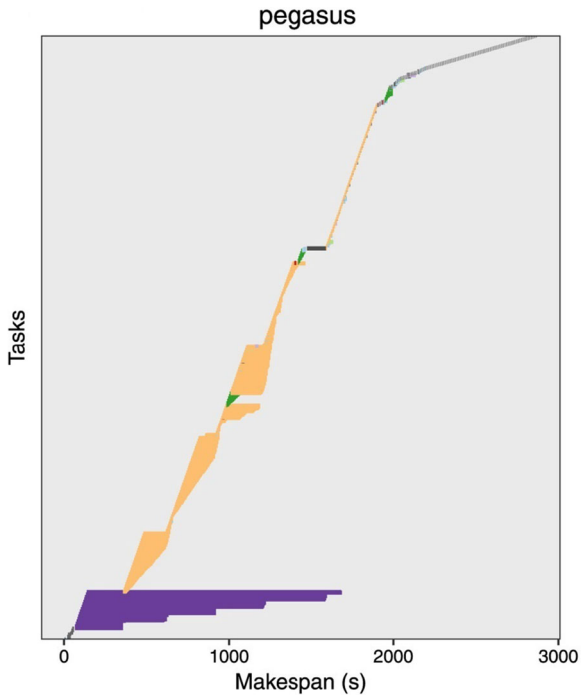


Fig. 14 Task execution Gantt chart for sample real-world ("pegasus") execution of the Montage-2.0 workflow on the AWS-m5.xlarge platform [14]

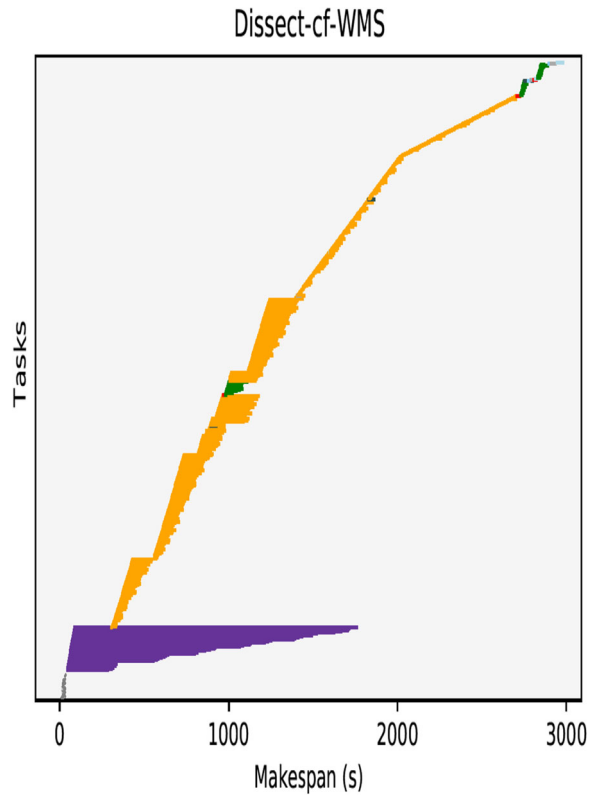


Fig. 15 Task execution Gantt chart for simulated Dissect-cf-WMS executions of the Montage-2.0 workflow on the AWS-m5.xlarge platform

times. Different kinds of tasks (executables) are indicated by different colours. We have tried to assign the same colours to these task types as in the real execution. The average time for real-world execution was 2911.8 seconds, whereas the average time for simulated execution was 2980 seconds. The results of the experiment demonstrate that the scheduling and execution of the simulated workflow are similar to the actual workflow execution. The runtime difference of 68.2 seconds is due to the errors of the prediction service utilised for choosing the activity and file transfer execution timings in the simulation, which inaccuracy is within a 3% range.

All tasks of the same type in this workflow have the same priority and are independent. For example, the shapes of the yellow areas differ in the two figures. The implementation-based behaviour of the workflow scheduler explains these differences. During the execution of the workflow, it is often possible to select several ready tasks for execution, e.g., groups of independent tasks on the same workflow level. If the number of computing resources, n, is less than the number of ready tasks, the scheduler immediately executes n ready tasks. In the majority of WMSs, these tasks are selected from the first n tasks returned during iteration through the data structures in which the task objects are placed. To create an identical replica of a WMS, you would need to develop and use the same data structures as the real implementation. Depending on the data structures, languages, and/or libraries used, this can be tedious or impossible. In this Pegasus case study, the real DAGMan scheduler uses a custom priority list to hold ready tasks, while our simulation version stores workflow tasks in a Java hashmap indexed by task string IDs. The consequence of this is that the real scheduler, when selecting the first n ready tasks, generally selects different tasks than the simulated version of the scheduler. The differences that can be seen in Figs. 14 and 15 can be attributed to this factor.

4.4 Auto-Scaling Mechanism

We focused on demonstrating the benefits of the auto-scaling mechanisms behind DISSECT-CF-WMS. We re-ran our large-scale (15K tasks) montage workflow. We used the same cloud we mentioned earlier. We compared the dynamic VM allocation strategies of the different auto-scalers with the completely static virtual infrastructure allocation (thus allowing a comparison to the previously acquired statically allocated Workflowsim Scenario). In the static scenario, we set up 50 virtual machines with 2 cores each in advance of the workflow and kept all VMs till the end. For this experiment, we used FirstFitScheduler as the VM scheduler, MPMC as the PM scheduler, and the DataDependency algorithm as the task scheduler. DISSECT-CF also simulates a single repository for a specific type of virtual appliance from which all VMs can be derived. The virtual appliance repository can significantly reduce the time required to create virtual machines. In this experiment, we modified the Pooling VI to ensure the efficiency of the auto-scaling mechanism. First, we adjusted the pooling VI to have 50 VMs with 2 cores each at the beginning of the workflow execution, since Montage has 12,495 tasks in the first and second phases. Second, we set the threshold for pooling VI to 80 VMs to reduce the cost while maintaining the makespan. Finally, the number of VMs is reduced to two if the single-threaded tasks of the Montage workflow are executed sequentially. In this case, one VM is used while the second is idle, because Pooling VI is designed to have a certain number of completely unused VMs available for executable jobs.

Figure 16 shows the results of executing the workflow. Pooling VI has the shortest total execution time in contrast to a dedicated cluster and the other of the auto-scaled virtual infrastructures. In terms of VM resource utilisation, Fig. 17 also shows that pooling VI has the best average VM utilisation across all mechanisms and static 50 VMs. This is because pooling VI has been configured to always keep one VM ready in the virtual infrastructure (so this is a compromise between the fully static and dynamic scenarios that the others implement). It is also worth noting that Pooling VI follows an almost static allocation of VMs, while the other two approaches frequently destroy and recreate VMs (in fact, they only reuse VMs for about six tasks before discarding them). These approaches thus significantly increase execution time, as most workflow tasks initially have no VMs to execute and must wait for their respective VMs to come to life. It should also be noted that the additional transfers required for staging data also lengthen execution, unlike the static VM allocation approach. In addition, the VMs access the same central storage to read and write the data dependency files. Montage is a data-intensive scientific workflow. However, the network bandwidth

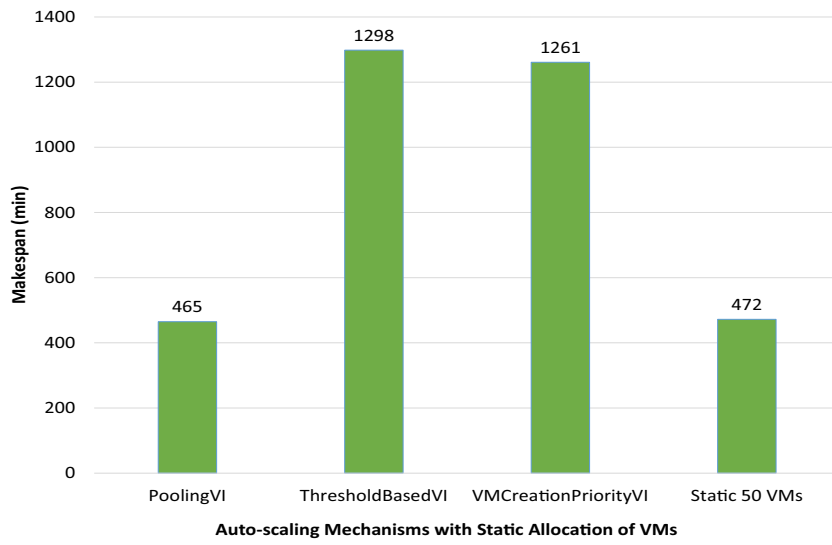


Fig. 16 Makespan of auto-scaling mechanisms and static 50 VMs with a large-scale Montage workflow (15000 tasks)

is fast enough to avoid bottlenecking the tasks that access the same central storage to store and retrieve the data files. Montage data sizes range from 4 to 1031 MB, with most of them being 4.2 MB, but a few are over 4.2 MB.

A similar concept to Amazon EC2 is being considered, where VMs are rented on demand and charged on an hourly basis, with partial hours rounded up to the next full hour. We find that Pooling VI reduced total billed hours by 41.5% compared to the dedicated

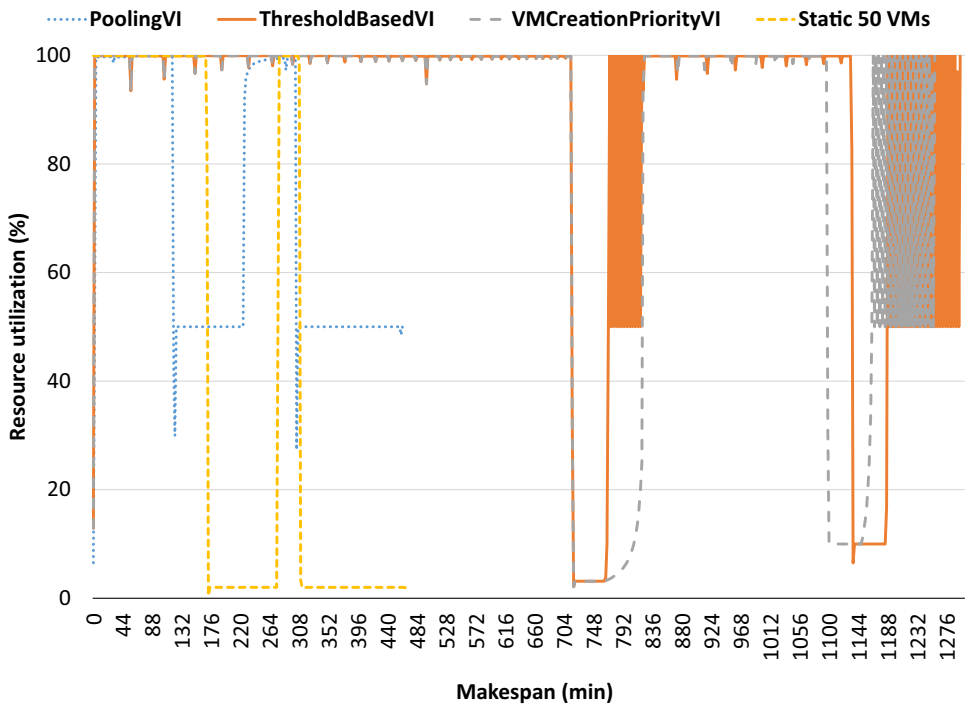


Fig. 17 Resource consumption patterns of auto-scaling mechanisms and static 50 VMs with a large-scale Montage workflow (15000 tasks)

cluster with 50 VMs, as shown in Fig. 18. The Montage workflow consists of six single-threaded tasks executed sequentially, with a total execution time of about 4.5 hours. As a result, when VMs were statically allocated, only one VM was used for 4.5 hours, while the other VMs were idle due to the single-thread tasks. Another consideration was related to Pooling VI, which describes the ability of mechanisms to allocate a number of VMs efficiently. Static provisioning is inefficient when the number of VMs remains constant over time. In this case, the scheduling algorithm does not provide a way to increase or decrease the number of VMs in response to a dynamic workload of workflows. In Fig. 19, Pooling VI reduced energy consumption by more than 82% compared to static allocation. In addition, Pooling VI is reduced by about 54% compared to the other auto-scaling mechanisms. Although Pooling VI has used the largest total number of VMs compared to static allocation and the other mechanisms (see Figs. 20 and 21), it has the lowest total number of hours billed, as shown in Fig. 18.

4.5 Scheduling Experiments

Finally, we compared the three aforementioned scheduling algorithms on DISSECT-CF-WMS without considering data transfers between VMs. We tested the algorithms on the same cloud that we defined earlier. We tested the three algorithms with the workflow applications on ten VMs. The VMs were heterogeneous in terms of the number of CPU cores from 1 to 10, with the first VM having one core while the last VM had ten cores. We collected the energy consumption of DISSECT-CF-WMS as shown in Fig. 22. We specifically note the need to collect energy consumption data for scheduling algorithms to evaluate their impact on energy consumption. We used the MPMC scheduler of the infrastructure and FirstFitScheduler as the VM scheduler. The algorithm HEFT has the best energy consumption over the other algorithms for all workflow applications except CyberShake, where MaxMin reduced energy consumption by 8% over HEFT. HEFT reduced energy consumption of LIGO by 45% and 50% over MaxMin and Minmin, respectively. Also HEFT has reduced the energy consumption of Sipht by 9% and 10% compared to MaxMin and MinMin respectively.

5 Conclusion and Future Works

A scientific workflow application consists of a large number of dependent jobs with complex priority constraints between them. Cloud workflow simulators do not currently provide sufficient support for the underlying virtualised infrastructure. This includes physical machine state scheduling, virtual machine creation details and virtual machine placement. Other simulators are often user-centric and treat the cloud as a black box. Unfortunately, this behaviour prevents the assessment of the impact on the infrastructure of the various decisions made by the WMS. In this

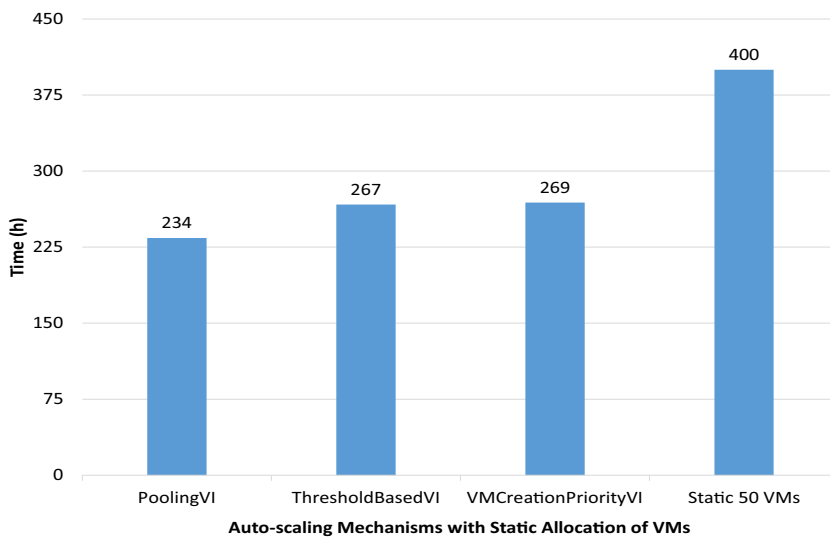


Fig. 18 The total accounted for hours of virtual machines of auto-scaling mechanisms and static 50 VMs with a large-scale Montage workflow (15000 tasks)

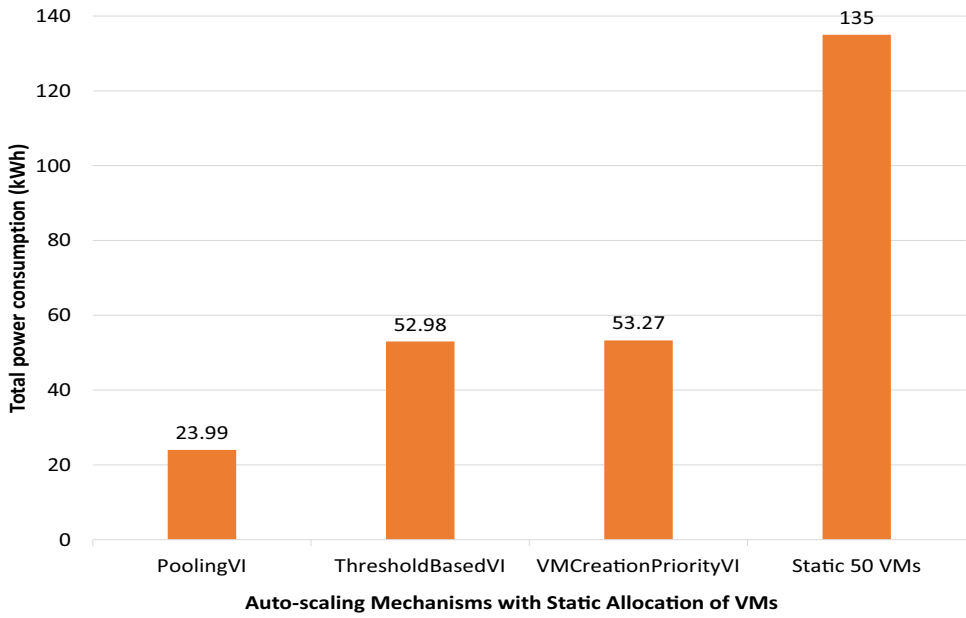


Fig. 19 The total power consumption (kWh) of auto-scaling mechanisms and static 50 VMs with a large-scale Montage workflow (15000 tasks)

paper, we present DISSECT-CF-WMS, a workflow management system simulation built on DISSECT-CF. We developed DISSECT-CF-WMS to focus on the user-side behaviour of the clouds, while DISSECT-CF focuses on the internal behaviour of the IaaS systems. It enables better energy awareness by allow-

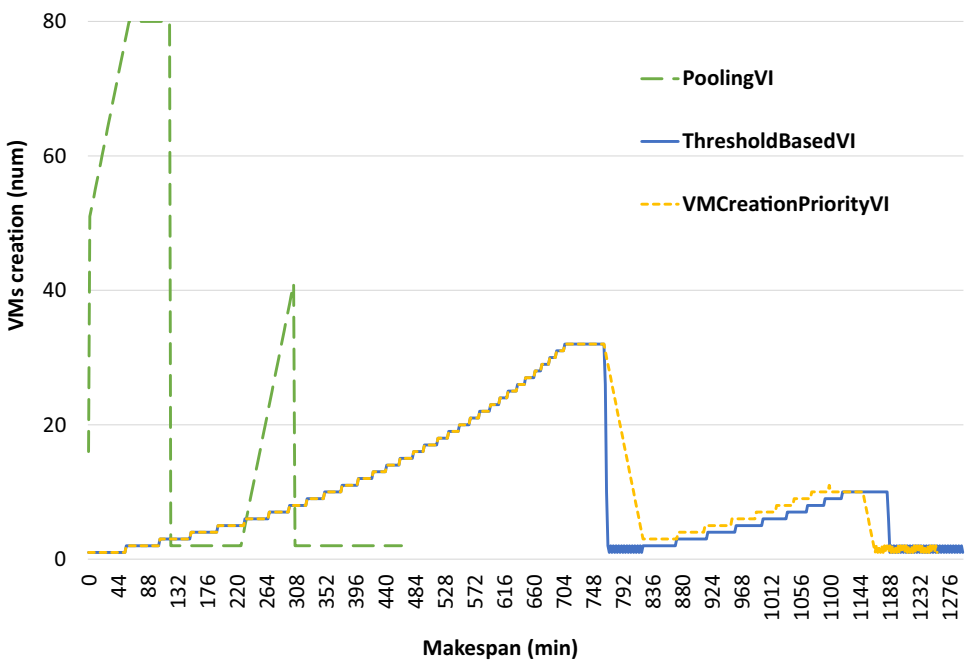


Fig. 20 The virtual machines creation patterns of auto-scaling mechanisms with a large-scale Montage workflow (15000 tasks)

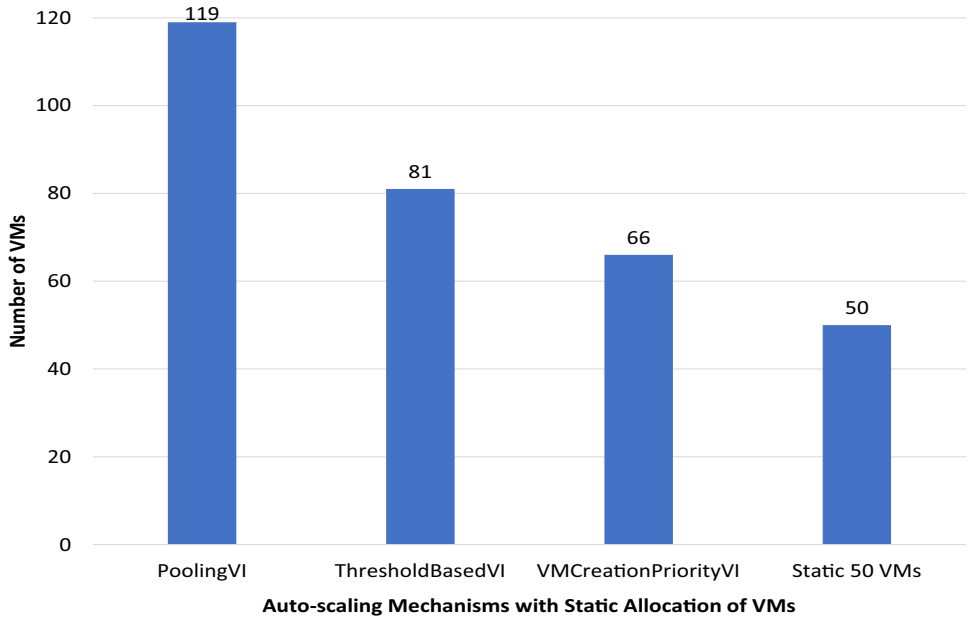


Fig. 21 The total accounted for the number of virtual machines (2 cores each) of auto-scaling mechanisms and static 50 VMs with a large-scale Montage workflow (15000 tasks)

ing investigation of physical machine schedulers and customisable consumption characteristics. It also provides dynamic provisioning to meet the resource needs of the workflow application as it runs on the infrastructure, taking into account the provisioning delay of a VM in the cloud. We evaluated our simulator by

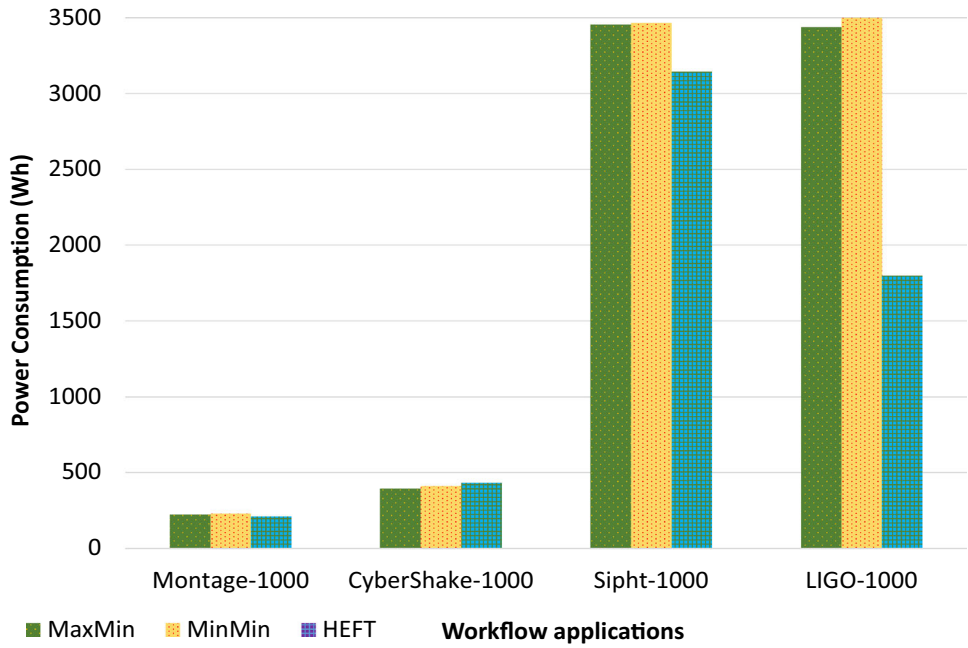


Fig. 22 The total power consumption of four workflow applications with three scheduling algorithms of DISSECT-CF-WMS on heterogeneous VMs

running several workflow applications with different schedulers of physical machines for a given infrastructure. The experimental results show that workflow researchers can investigate different PM schedulers on infrastructure configurations to achieve lower energy consumption. The experiments also show that DISSECT-CF-WMS is up to 295× faster than WorkflowSim and still delivers accurate results. The experimental results of the auto-scaling mechanism show that the integration has the potential to optimise makespan, energy consumption, and VM utilisation over static provisioning. This work also allowed us to investigate Internal IaaS behavioural knowledge, such as different scheduling strategies for physical machines in a simulated environment; DISSECT-CF-WMS proved very useful.

In the future, we will further extend the DISSECT-CF-WMS scheduling algorithm for dynamic provisioning so that we can consider cost, makespan, resource utilisation and energy consumption simultaneously. Multi-objective optimisation is a hot research area in workflow scheduling. Users can create algorithms for multi-objective scheduling optimisation by leveraging their knowledge of IaaS internals. These insights can be used to optimise key workflow objectives (such as energy consumption, time and resource utilisation). DISSECT-CF-WMS offers opportunities for scientific workflow applications that can be used for the upcoming research areas:

Resource Usage. The new feature of DISSECT-CF makes it possible to identify different physical machines (PMs) for instantiated VMs. To gain insight into WMS usage in private cloud deployments, workflow schedulers can collect information from PM schedulers and VM placement mechanisms. This enables an accurate knowledge of VMs sharing the same PM. Therefore, workflow schedulers can schedule tasks with high data dependency files for these VMs. As a result, these VMs can reduce data transfer times because they share the same PM or are close to each other (in terms of the network). In addition, different allocation strategies of PMs to VMs can be developed to study their impact on performance, resource utilisation, energy consumption and fairness of workflows. More mechanisms could be added to reflect the environment in real life. An open research topic that cannot be discussed with commercial cloud companies because their VM placement technique is not public is the impact of background load on virtual resource performance. When there are multiple instances on a physical machine, performance degradation is always possible. If the network, memory or CPU become bottlenecks, virtual machine performance can be affected by this behaviour.

Power Consumption. Energy awareness is very much needed to support energy-efficient data centres. It is not enough to focus on data centre operators when we are looking at energy efficiency. It is important that our applications that use these data centres are also energy-aware. To enable energy awareness for workflow schedulers, measurements provided by DISSECT-CF can lead to optimisation of workflow execution by considering energy consumption alongside traditional cost and time objectives. These alternatives, which are closer to energy-optimal plans, increase the potential to reduce energy consumption and lower data centre operating costs.

The use of the underlying PM and its resources has a greater impact on energy consumption. For example, data centres can reduce the perceived energy consumption of a virtual CPU for customers who are under-utilising their VMs by allocating fractions of real CPUs from the PMs hosting those VMs. For example, some PM schedulers from DISSECT-CF are able to shut down machines when the VM queue is empty and there are some physical machines with no load. Therefore, this behaviour in cloud systems could be exploited in future WMSs by optimising energy consumption.

Data Centre Configurations. The DISSECT-CF simulator allows the properties of data centres (DCs) to be specified using the CloudLoader class. For our workflow executions, we can define both homogeneous and heterogeneous computing resources. Therefore, the base simulator allows evaluating different types of data centres to find out which could be the best option for a specific type of workflow application. This allows the impact of the DC configuration on scientific workflow applications to be investigated through a series of experiments. The rest of the DISSECT-CF-WMS configuration remains the same, but the DC features change from one experiment to the next. DISSECT-CF-WMS provides a federated

cloud that transparently aggregates different cloud computing providers. This allows users to use the federated cloud by developing an approach to run workflows that are compute and data intensive applications. They have different requirements for the tasks that are suitable for execution on multiple IaaS cloud providers.

Background Load. Virtualization is an important component of IaaS because it isolates access to resources through virtual machines and allows users to share physical resources securely. While IaaS clouds offer some ability to manage a virtual ensemble of resources (called virtual infrastructures), they offer no way to accurately track the status, utilisation or performance of their resources. The physical layer is completely hidden. Due to the multi-tenant nature of clouds, application performance can be severely impacted by other, unknown and invisible processes, known as background workload. For this reason, DISSECT-CF-WMS can be used to add background load to a simulation by interfacing with the Grid Workload Archive file format [24]. If required, background load can be added to resources, either by using traces from real background loads, such as the Grid Workload Archive, or from synthetic workloads.

Performance Metrics. DISSECT-CF-WMS provides the following types of performance metrics, such as CPU utilisation, disc read and write throughput, incoming network data, outgoing network data and average queue time. They can be collected during workflow execution preiodically at a user-specified interval. This information helps to understand and compare the actual resource consumption during workflow execution under different conditions, e.g. with a different number of resources or using different resource configurations. We can also conclude from this information that the execution phases of a workflow are CPU -intensive, memory-intensive and I/O-intensive. Therefore, the observed metrics can help users when they need to execute their workflows in the real context of cloud computing, such as the number of resources, network bandwidth and memory size.

Funding Open access funding provided by University of Miskolc. This work was supported in part by the Hungarian Scientific Research Fund under Grant agreement OTKA FK 131793.

Data Availability The application developed in this manuscript is publicly available under an open source license at https://github.com/kecskemeti/dissect-cf-examples.

Compliance with Ethical Standards

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Abramovici, A., Althouse, W.E., Drever, R.W., Gürsel, Y., Kawamura, S., Raab, F.J., Shoemaker, D., Sievers, L., Spero, R.E., Thorne, K.S., et al.: LIGO: The laser interferometer gravitational-wave observatory. Science **256**(5055), 325–333 (1992)
2. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S.: Kepler: an extensible system for design and execution of scientific workflows. In: Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004, pp. 423–424. IEEE (2004)
3. Bell, W.H., Cameron, D.G., Millar, A.P., Capozza, L., Stockinger, K., Zini, F.: Optorsim: A grid simulator for studying dynamic data replication strategies. Int. J. High Perform. Comput. Appl. **17**(4), 403–416 (2003)
4. Benoit, A., Rehn-Sonigo, V., Robert, Y.: Optimizing latency and reliability of pipeline workflow applications. In: 2008 IEEE International Symposium on Parallel and Distributed Processing, pp. 1–10. IEEE (2008)
5. Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., Kennedy, K.: Task scheduling strategies for workflow-based applications in grids. In: CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005, vol. 2, pp. 759–767. IEEE (2005)
6. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput. **61**(6), 810–837 (2001)
7. Brown, D.A., Brady, P.R., Dietz, A., Cao, J., Johnson, B., McNabb, J.: A case study on the use of workflow technologies for scientific analysis: Gravitational wave data

analysis. In: Workflows for e-Science, pp. 39–59. Springer (2007)
8. Buyya, R., Murshed, M.: Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurr. Comput. Pract. Experience **14**(13-15), 1175–1220 (2002)
9. Cai, Z., Li, Q., Li, X.: Elasticsim: A toolkit for simulating workflows with cloud resource runtime auto-scaling and stochastic task execution times. J. Grid Comput. **15**(2), 257–272 (2017)
10. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Experience **41**(1), 23–50 (2011)
11. Cao, J., Jarvis, S.A., Saini, S., Nudd, G.R.: Gridflow: Workflow management for grid computing. In: CCGrid 2003. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003. Proceedings, pp. 198–205. IEEE (2003)
12. Casanova, H., Giersch, A., Legrand, A., Quinson, M., Suter, F.: Versatile, scalable, and accurate simulation of distributed applications and platforms. J. Parallel Distrib. Comput. **74**(10), 2899–2917 (2014)
13. Casanova, H., Pandey, S., Oeth, J., Tanaka, R., Suter, F., da Silva, R.F.: Wrench: A framework for simulating workflow management systems. In: 2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS), pp. 74–85. IEEE (2018)
14. Casanova, H., da Silva, R.F., Tanaka, R., Pandey, S., Jethwani, G., Koch, W., Albrecht, S., Oeth, J., Suter, F.: Developing accurate and scalable simulators of production workflow management systems with wrench. Futur. Gener. Comput. Syst. **112**, 162–175 (2020)
15. Chen, W., Deelman, E.: WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In: 2012 IEEE 8th International Conference on E-science, pp. 1–8. IEEE (2012)
16. Da Silva, R.F., Glatard, T., Desprez, F.: Self-healing of workflow activity incidents on distributed computing infrastructures. Futur. Gener. Comput. Syst. **29**(8), 2284–2294 (2013)
17. Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.H., Vahi, K., Livny, M.: Pegasus: Mapping scientific workflows onto the grid. In: European Across Grids Conference, pp. 11–20. Springer (2004)
18. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., Da Silva, R.F., Livny, M., et al.: Pegasus, a workflow management system for science automation. Futur. Gener. Comput. Syst. **46**, 17–35 (2015)
19. Garg, S.K., Buyya, R.: NetworkCloudSim: Modelling parallel applications in cloud simulations. In: 2011 4th IEEE International Conference on Utility and Cloud Computing, pp. 105–113. IEEE (2011)
20. Graves, R., Jordan, T.H., Callaghan, S., Deelman, E., Field, E., Juve, G., Kesselman, C., Maechling, P., Mehta, G., Milner, K., et al.: Cybershake: A physics-based seismic hazard model for southern california. Pure Appl. Geophys. **168**(3-4), 367–381 (2011)
21. Gu, Y., Wu, Q.: Maximizing workflow throughput for streaming applications in distributed environments. In: 2010 Proceedings of 19th International Conference on Computer Communications and Networks, pp. 1–6. IEEE (2010)
22. Hirales-Carbajal, A., Tchernykh, A., Röblitz, T., Yahyapour, R.: A grid simulation framework to study advance scheduling strategies for complex workflow applications. In: 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), pp. 1–8. IEEE (2010)
23. Hoefler, T., Schneider, T., Lumsdaine, A.: Loggopsim: simulating large-scale applications in the LogGOPS model. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, pp. 597–604 (2010)
24. Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., Epema, D.H.: The grid workloads archive. Futur. Gener. Comput. Syst. **24**(7), 672–686 (2008)
25. Juve, G., Deelman, E.: Resource provisioning options for large-scale scientific workflows. In: 2008 IEEE 4th International Conference on eScience, pp. 608–613. IEEE (2008)
26. Kandaswamy, G., Mandal, A., Reed, D.A.: Fault tolerance and recovery of scientific workflows on computational grids. In: 2008 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID), pp. 777–782. IEEE (2008)
27. Kecskemeti, G.: DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds. Simul. Model. Pract. Theory **58**, 188–218 (2015)
28. Kecskemeti, G., Ostermann, S., Prodan, R.: Fostering energy-awareness in simulations behind scientific workflow management systems. In: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp. 29–38. IEEE (2014)
29. Livny, J., Teonadi, H., Livny, M., Waldor, M.K.: High-throughput, kingdom-wide prediction and annotation of bacterial non-coding RNAs. PloS ONE **3**(9), e3197 (2008)
30. Malawski, M., Juve, G., Deelman, E., Nabrzyski, J.: Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. Futur. Gener. Comput. Syst. **48**, 1–18 (2015)
31. Milojičić, D., Llorente, I.M., Montero, R.S.: Opennebula: A cloud management tool. IEEE Internet Comput. **15**(2), 11–14 (2011)
32. Nunez, A., Vazquez-Poletti, J.L., Caminero, A.C., Carretero, J., Llorente, I.M.: Design of a new cloud computing simulation platform. In: International Conference on Computational Science and Its Applications, pp. 582–593. Springer (2011)
33. Ostermann, S., Kecskemeti, G., Prodan, R.: Multi-layered simulations at the heart of workflow enactment on clouds. Concurr. Comput. Pract. Experience **28**(11), 3180–3201 (2016)
34. Ostermann, S., Plankensteiner, K., Bodner, D., Kraler, G., Prodan, R.: Integration of an event-based simulation framework into a scientific workflow execution environment for

grids and clouds. In: European Conference on a Service-Based Internet, pp. 1–13. Springer (2011)
35. Ostermann, S., Plankensteiner, K., Prodan, R.: Using a new event-based simulation framework for investigating resource provisioning in clouds. Sci. Program. **19**(2-3), 161–178 (2011)
36. Ostermann, S., Plankensteiner, K., Prodan, R., Fahringer, T.: GroudSim: An event-based simulation framework for computational grids and clouds. In: European Conference on Parallel Processing, pp. 305–313. Springer (2010)
37. Ostermann, S., Prodan, R., Fahringer, T.: Dynamic cloud provisioning for scientific grid workflows. In: 2010 11th IEEE/ACM International Conference on Grid Computing, pp. 97–104. IEEE (2010)
38. Tikir, M.M., Laurenzano, M.A., Carrington, L., Snavely, A.: Psins: An open source event tracer and execution simulator for MPI applications. In: European Conference on Parallel Processing, pp. 135–148. Springer (2009)
39. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
40. Tsai, M.H., Lai, K.C., Chang, H.Y., Chen, K.F., Huang, K.C.: Pewss: A platform of extensible workflow simulation service for workflow scheduling research. Softw. Pract. Experience **48**(4), 796–819 (2018)
41. Ullman, J.D.: Np-complete scheduling problems. J. Comput. Syst. Sci. **10**(3), 384–393 (1975)
42. Vydyanathan, N., Catalyurek, U.V., Kurc, T.M., Sadayappan, P., Saltz, J.H.: Toward optimizing latency under throughput constraints for application workflows on clusters. In: European Conference on Parallel Processing, pp. 173–183. Springer (2007)
43. Zheng, G., Kakulapati, G., Kalé, L.V.: Bigsim: A parallel simulator for performance prediction of extremely large parallel machines. In: 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings, p. 78. IEEE (2004)