CWOM: a lightweight cloud-oriented workflow optimisation middleware

Peng Xiao

School of Computer and Communication,
Hunan Institute of Engineering,
Xiangtan City, Hunan Province 411104, China
Email: efn4623@126.com

Abstract: In recent years, more and more workflow applications have been developed to execute on various kinds of cloud platforms. Although many cloud-based workflow management and enactment systems have been proposed, there are still many challenges when deploying and executing workflows on various kinds of real-world cloud platform. In this paper, we present a lightweight framework, namely *cloud-oriented workflow optimisation middleware* (CWOM), which is designed for easing the work of optimising the workflow execution efficiency as well as the work of realising fine-grained workflow management. The proposed CWOM is implemented as a plug-in middleware that can interact with the other workflow services and therefore provide more desirable services for workflow applications. Another advantage of our CWOM is that its services are loosely coupled with other cloud services, which makes it more flexible and configurable in most of the real-world cloud platforms. To investigate the effectiveness of the proposed system, we deploy the prototype implementation of CWOM in a campus-based cloud platform, and the experimental results indicate that it can significantly improve the delivered QoS and enhance the service level offered by conventional workflow solutions.

Keywords: cloud computing; workflow; virtual machine; performance monitor.

Reference to this paper should be made as follows: Xiao, P. (2021) 'CWOM: a lightweight cloud-oriented workflow optimisation middleware', *Int. J. Networking and Virtual Organisations*, Vol. 24, No. 1, pp.67–83.

Biographical notes: Peng Xiao received his PhD from the Central South University, in 2009. Currently, he works in the Hunan Institute of Engineering as an Associated Professor. His research interests include cloud computing, distributed resource management, networking optimisation, and data centre energy-efficiency optimisation. He is a senior member of the CCF and member of IEEE Computer Society.

1 Introduction

Cloud computing has emerged as a promising IT-infrastructure for deploying and running the next-generation of e-science applications (Aslam et al., 2017; Weerasiri et al., 2017). Comparing with other distributed paradigms (i.e., cluster and grid), cloud systems has some significant advantages including cost-effective, elastic provisioning, isolated execution, etc. (Sookhak et al., 2014; Rodriguez and Buyya, 2017). Consequently, more and more non-trivial applications have been developed to execute on various kinds of cloud platforms. Among these applications, workflow is one the most important types especially when plenty of as computational devices, data, applications, and scientific instruments are involved and need to be orchestrated (Wu et al., 2015; Xiao and Hao, 2016; Casas et al., 2017). Imposing the workflow paradigm for application composition offers several advantages, including orchestrated resource sharing, optimised resource utilisation, automatic data processing, inter-organisational collaboration and so on (Zhao et al., 2015; Qi, 2017). Therefore, scientific communities, such as high-energy physics, bioinformatics, geophysics, etc., have widely applied workflow to perform complex simulations or process large volumes of datasets (Le-Blanc et al., 2013; Krieger et al., 2017).

Generally saying, a workflow application is concerned with a set of automatic procedures, in which datasets are processed based on a pre-defined set of rules with aiming at obtaining certain objectives (Deelman et al., 2015). According to the definition of workflow management coalition, a workflow management system should be capable of defining, managing, executing user's workflows on a group of networking resources (Casas et al., 2017). Due to the importance of workflow paradigm in distributed systems, plenty of workflow systems, platforms, services or techniques have been proposed in the past decades (Marinho et al., 2012; Zeng et al., 2015; Bryk et al., 2016; Khaleel and Zhu, 2016; Abdi et al., 2018). In those previous works, earlier works mainly focused on conventional high-performance computing platforms such as clusters and grids, while the recent works mainly took efforts on how to deploy and execute workflows in cloud environments. By summarising the recent studies, the challenges of deploying and executing workflows on cloud platforms are as following:

1 Unpredictable workloads on shared resources are likely to reduce the user's QoS satisfactory (Moens and De-Turck, 2015; Khaleel and Zhu, 2016).

2 Prolonged execution times of workflows tend to introduce more failures on various kinds of resources, which in turn increases the complexity of managing workflow life-cycle (Yun et al., 2015; Abdi et al., 2018).

3 Different kinds of workflows may require distinctive level of workflow services, which may be quite simple or complicated (Xiao and Hao, 2016; Rociszewski et al., 2016).

In this study, we present a lightweight framework, namely *cloud-oriented workflow optimisation middleware* (CWOM), which is designed for easing the work of optimising the workflow execution efficiency as well as the work of realising fine-grained workflow management. Rather than replacing the conventional workflow engines or enactment systems, the proposed CWOM is implemented as a plug-in middleware that can interact with the other workflow services and therefore provide more desirable services for workflow applications. In the CWOM framework, we implement three workflow services, including workflow definition script, SLA matching service and execution coordinating service. With the help of the CWOM's services, the conventional workflow engine cannot only improve the delivered QoS but also provide more flexible and desirable managing service when running user's workflow applications. Another advantage of our CWOM is that its services are loosely coupled with other cloud services, which makes it more flexible and configurable in most of the real-world cloud platforms.

The rest of this paper is organised as follows. Section 2 presents the related work. In Section 3, we describe the framework of CWOM and the corresponding implementations of the key services. In Section 4, experiments are conducted to investigate the effectiveness of the CWOM framework. Finally, Section 5 concludes the paper with a brief discussion of the future work.

2 Related work

Workflow management has attracted plenty of attentions for a long time in distributed computing and processing area. In early studies, workflow-oriented systems were mainly developed for high-performance clusters or heterogeneous grids, where IT infrastructures are organised a set of common resource pool and offers physical resources for running user's applications (including workflows). Many workflow systems were designed for deal a specific aspect. For example, DAGMap (Cao et al., 2010) is a simple and efficient workflow scheduling framework, which divides the workflow scheduling procedure into two phases: static mapping and dependable execution. When using the DAGMap framework, sub-task grouping is based on dependency relationships between tasks, and the min-min and min-max scheduling policies were used for scheduling independent tasks. Also, the DAGMap framework provides a simple checkpoint service to ensure the dependable execution of workflows. BTS (Byun et al., 2011) is a prediction-based resource broker system for workflows, which is able to estimate the minimum number of physical hosts required to complete workflows before the pre-defined deadline of users. The idea of BTS has been widely adopted by other studies when soft/hard deadline measurement of applications is underlined. P-GRADE (Farkas and Kacsuk, 2011) is an open-source portal system, which support creation, execution and management of workflows in grid environments. Currently, the P-GRADE portal service is still used in many cloud-oriented workflow systems. ProvManager (Marinho et al., 2012) is a provenance management framework which can reduce the overheads when gathering and analysing the provenance information of workflow applications that running in a distributed and heterogeneous environment. The service offered by ProvManager was very useful when multiple workflows are concurrently deployed in a single distributed system. DataFlow (Zeng et al., 2015) is an independent dataflow analysing model, which can be used to reveal the data-dependency among different tasks and therefore provide information for making more suitable decisions on resource allocating and task scheduling.

In recent years, how to deploy and execute workflow applications in cloud environments has become a hot topic, and plenty of studies have taken efforts on developing cloud-oriented workflow services or systems. For example, HSGA (Delavar and Aryan, 2014) is a hybrid workflow scheduling framework which applies genetic algorithm to improve different performance metrics (i.e., responsive time, load balancing, reliability, etc.) of workflows that running in virtualised cloud platforms. The key ideology of HSGA is that it merges best-fit and Round-Robin policies together so as to ensure the initial population is optimal and a good solution can be quickly obtained. CometCloud (Diaz-Montes et al., 2015) is a famous project that aims to provide infrastructure and programming support for enabling application workflows in cloud-like systems. Currently, the CometCloud system enables flexible software-defined synthesis through the on-demand federation of geographically distributed resources. WaFS (Wang et al., 2015) is a cloud-oriented file-system which can automatically detect and gather the explicit and implicit data dependencies between workflow jobs. When accessing the data in WaFS, a workflow scheduler can either make effective cost-performance trade-offs or improve storage utilisation. SideIO (Wang et al., 2017) is a simple but efficient data-migration tool, which consists of three components, including I/O splitter, I/O middleware and I/O scheduler. By calling these components in SideIO, data-intensive workflows are able to achieve very desirable read/write I/O performance regardless of the intensiveness of up-level workloads on storage servers.

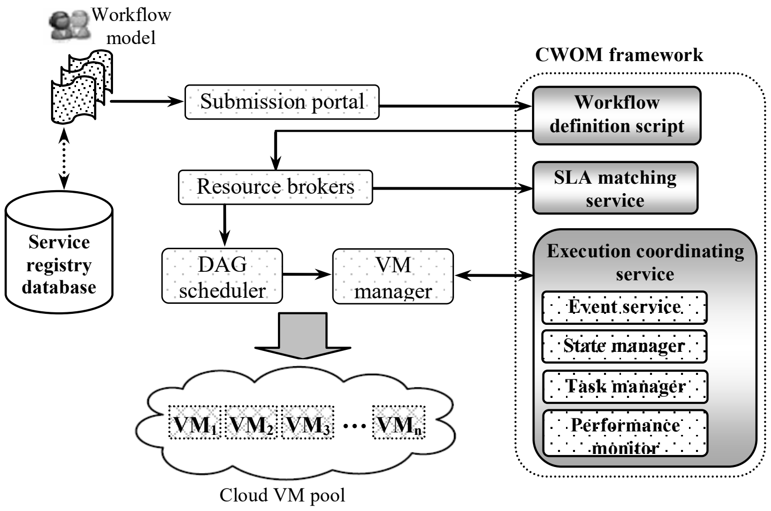
Besides the above workflow systems, many researchers also proposed many solutions to optimise the execution efficiency, dependability, energy-efficiency of workflow applications in cloud environments. For example, Bryk et al. (2016) implemented a simulation framework for deal with file transfers between sub-tasks, with aiming at featuring the ability to dynamically allocate bandwidth and supporting a configurable data replication schema. Khaleel and Zhu (2016) addressed the energy-consumption problem of large-scale workflows that run in cloud systems. Based on a rigorous mathematical model, they proposed a multiple-procedure heuristic workflow scheduling and consolidation strategy to maximise the resource utilisation and minimise the power consumption. Cai et al. (2017) proposed a dynamic resource provisioning and scheduling algorithm which is able to satisfy the workflow deadline requirement by using the sum of sub-task execution time expectation and standard deviation to estimate actual task execution time. According to their experimental results, such a scheduling algorithm is very effective for those workflows with relative stable makespan. Rimal and Maier (2017) presented a workflow scheduling (CWSA) policy for compute-intensive workflow applications in multi-tenant cloud computing environments. The CWSA policy is able to minimise the overall workflow completion time, tardiness, cost of execution of the workflows, and utilise idle resources of cloud effectively. Abdi et al. (2018) proposed a financial cost model which can be used to minimise the resource costs of bag-of-tasks (BoT) workflows. Based on this a financial cost model, the resource allocation problem can be formulated as an integer linear programming problem and approximately solved by a greedy randomised adaptive search procedure. Luo et al. (2018) applied the epidemic theory to model the dynamics of time delay propagation when deploying and running large-scale workflow applications. This proposed temporal violation transmission model can estimate the account of temporal violations and determine the account of violations that must be handled.

3 Cloud-oriented workflow optimisation middleware

3.1 Framework overview

In a typical cloud system, a set of VM instances are provided for running various kinds of user applications. To deploy and execute a workflow, there are four indispensable services, including submission portal, resource broker, DAG scheduler and VM manager. The proposed CWOM framework works as a plug-in middleware, which is aiming at providing a fine-grained workflow management and enhancing the user's QoS satisfactory. The overview of CWOM framework in a typical cloud environment is demonstrated in Figure 1.

Figure 1 The framework overview of CWOM in a virtualised cloud environment

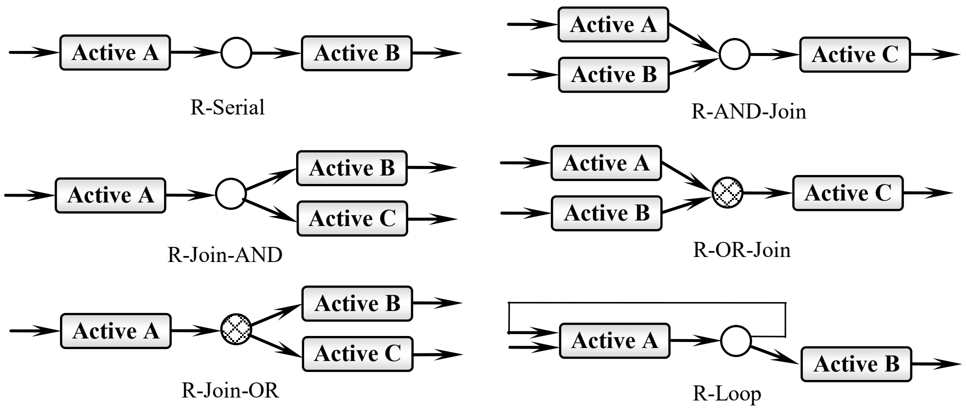


In the CWOM framework, we implement six well-defined services, including workflow definition script, SLA matching service, event service, state manager, task manager and performance monitor. Based on their working logics, the later four services are implemented together and therefore integrated as the execution coordinating service. The workflow definition script service is responsible for translating the user's workflow model as a set of XML specifications that can be used for resource allocating and task scheduling. The SLA matching service is responsible for finding a set of suitable resources which has the potential of minimising the deployment and satisfying the QoS requirements during the execution. As to the execution coordinating service, it is introduced to help other workflow/cloud services to deal with the issues that occur after scheduling decisions, such as exception handling and workflow state managing. With the help of the CWOM's services, the conventional workflow engine cannot only improve the delivered QoS but also provide more flexible and desirable managing service when running user's workflow applications. As an independent middleware, the proposed CWOM middleware is loosely-coupled with those existing cloud infrastructures, and interacts with them only through well-defined interfaces. In the following sections, we present the designing and implementation of the CWOM's services, and describe how it works with other cloud/workflow services.

3.2 Workflow definition based on rules

In existing workflow platforms, defining the abstractions of a workflow is often completed by using certain standalone toolkit, which can be categorised into three types including visualised graphic tool, scripting language, or combination of both. Typically, visualised graphic tools are favoured by common users due to their intuitive feature, while the scripting-based tools are more welcomed by skilled users since they provides more flexible and fine-grained controlling for defining the abstractions of workflows. In our CWOM framework, we introduce a novel scripting-based definition mechanism which is based on a set of pre-defined basic rules. By using these rules, any user can quickly define the abstraction of a workflow through a XML file. Also, some automatic tools can be used to compose multiple workflows into a single one based on this rule-based scripting language.

Figure 2 The rules and their corresponding workflow patterns



Before present the defining rules, we first categorise the abstractions of any workflow as four kinds of components, including *activity*, *logic operation*, *flow* and *dataset*. The activity components are defined as the subtasks of a workflow, and there are at least two special activities (init_activity and exit_activity) presenting the beginning and ending of a workflow model. An *activity* can be defined as a mathematic function or a complicated workflow, which is used to specify the services that users want to invoke. The *logic* components are used to control the logic relationships among various kinds of components. Currently, we defined four kinds of logic operations: AND-AND, OR-OR, AND-OR and OR-AND. For each logic operation, it is connected with a trigger event and a set of ruling branches. The *flow* components are used to specifying the communications between different activities or the control of services. The *dataset* components are used to defining the input/output information during the execution of a workflow. Based on the above components, we can define the following rules, including R-Serial, R-AND-Join, R-AND-Split, R-OR-Join, R-OR-Split, R-Loop, which basically cover the abstraction patterns of the most of workflow models. In Figure 2, we demonstrate these rules as well as the corresponding workflow patterns. Based on the above rules, the definition of a workflow can separated as a set of rules and formally described through a XML file, which can be demonstrated as the following example.

```
<?xml version="1.0" encoding="UTF-8" ?>
<s:WFD name="example_workflow">
<r:R-Serial>
    <activity dataset="in.db">init_activity</activity>
    <flow>1</flow>
    <logic_op>OR</logic_op>
    <activity>A_activity</activity>
    <activity dataset="B.txt">B_activity</activity>
</r:R-Serial >
<r:AND-Join>
    <activity>A_activity</activity>
    <activity>B_activity</activity>
    <flow>0</flow>
    <logic_op>AND</logic_op>
    <activity dataset="out.db">C_activity</activity>
</r:AND-Join >
<r:R-Serial>
    <activity dataset="in.db">C_activity</activity>
    <flow>1</flow>
    <logic_op>AND_AND</logic_op>
    <activity>exit_activity</activity>
</r:R-Serial>
</s:WFD>
```

3.3 SLA matching service

The goal of SLA matching service in CWOM is to find a set of suitable resources which has the potential of minimising the deployment and satisfying the QoS requirements during the execution. To deal with this problem, we introduce a simple but effective matching model which can be used to complete a series of resource matching request in a batch manner. At first, we select the most-mentioned QoS metrics, including throughput noted as $Q^{th}(x)$, resource availability noted as $Q^{av}(x)$, responsive time noted as $Q^{rt}(x)$, and execution latency noted as $Q^{el}(x)$. Therefore, for a given cloud resource x, its aggregated SLA values can be denoted as $Q(x) = \{Q^{th}(x), Q^{av}(x), Q^{rt}(x), Q^{el}(x)\}$. From the perspective of cloud users, a SLA request can be similarly denoted as $Q(r) = \{Q^{th}(r), Q^{av}(r), Q^{rt}(r), Q^{el}(r)\}$, which denotes the minimum or maximum SLA values that can be accepted by users. For any SLA matching schema, the resource-related cost can be formulated as:

$$C^{res}(M) = \sum_{i=1}^{n} p_i^{vm} \sum_{t_j \in \Omega(vm_i)} et\left(vm_i, t_j\right) \tag{1}$$

where M is the resource allocation scheme which is often denoted as $m \times n$ matrix, p_i^{vm} is the price of vm_i, t_j is the sub-task in the workflow, $et(vm_i, t_j)$ is the execution time when t_j is assigned onto the vm_i for running. Based on the above definitions, we formulate the SLA matching problem as:

$$\begin{cases} \max\left\{Q^{th}(x), Q^{av}(x)\right\}, \text{ where } Q^{th}(x) \geq Q^{th}(r) \cap Q^{av}(x) \leq Q^{av}(r) \\ \min\left\{Q^{rt}(x), Q^{el}(x), C^{res}(M)\right\}, \text{ where } Q^{rt}(x) \geq Q^{th}(x) \cap Q^{el}(x) \leq Q^{el}(r) \end{cases} \tag{2}$$

Generally saying, the SLA matching problem is NP-hard as there are multiple resource candidates and multiple SLA attributes at the same time. So, we apply the attribute auction model (AAM) to find a most suitable solution for the SLA matching problem (2). The AAM theory was introduced in economics at first with aiming at solving the problem of multi-attribute online auction. The key ideology of AAM is to use a quasi-linear utility function to maximise the user profit for the given SLA level. To achieve this goal, the SLA preferences of all users are required to be translated into weighted scoring functions, even for those non-functional SLA attributes. For example, the cost function in (XX) will be translated as:

$$C^{res}(R \mid M) = \sum_{r_i \in R} c^{res}(r_i) \cdot \omega(r_i) - C^{res}(M) \tag{3}$$

where $c^{res}(r_i)$ is the willingness of a user who likely to pay for a given request r_i, $\omega_i(r_i)$ is the scoring function that translates the SLA level into a relative fulfilment level of user's requirements, which can be defined as following:

$$\omega(r_i) = \sum_{j=1} w_i(a_j) \cdot fitness(a_j) \tag{4}$$

where $w_i(a_j)$ is the relative weight for a given SLA attribute a_j, $fitness(a_j)$ is the fitting function that the monitored SLA values to a normalised value in [0, 1].

Based on the AAM theory, the SLA matching algorithm is quite straightforward, which mainly is consisted of five steps:

1 Transform the abstracted workflow model into XML-based doc.

2 Select out the candidate providers based on auction biding results.

3 Use list-scheduling algorithm to perform SLA matching and evaluate the obtained SLA values for the current request.

4 Assign the request onto the target VM instances.

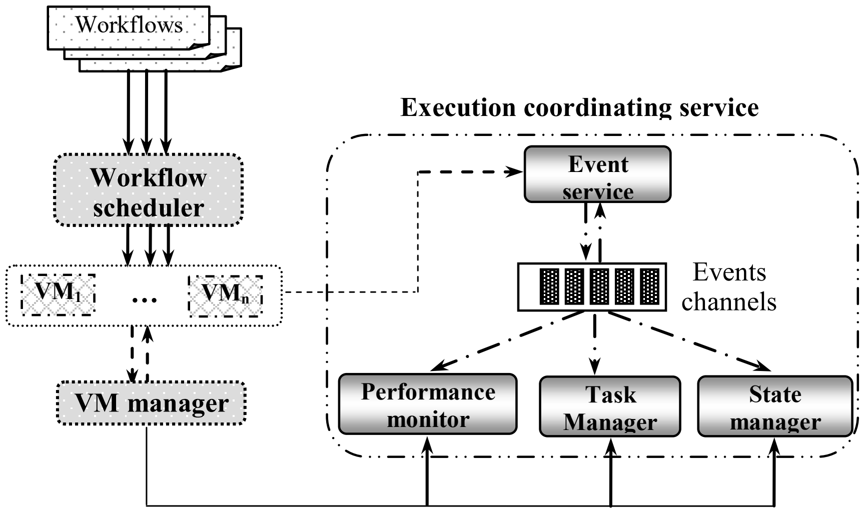
5 Update the runtime information of the target VM by SLA monitoring service.

In the above steps, we use the heuristic list-scheduling to implement the Step 3 with aiming at reducing the complexity of the service's implementation. In fact, it can be replaced by other intelligent algorithms, such as evolutionary algorithm or genetic algorithm if more optimal results are required.

3.4 Execution coordinating service

During the execution of workflows, many kinds of events might occur and some of these events may have significant effects on the running efficiency or final result of the target workflows. For instance, allocated resources might be unavailable at runtime or communications between different tasks might be blocked due to the intensive I/O traffics. In our CWOM, the *execution coordinating service* is designed to deal with the complexities of running workflows in cloud environments. It is noteworthy that the execution coordinating service does not perform the traditional task scheduling operations. In fact, it is used to help the traditional scheduler to deal with those issues that occur after scheduling decisions, such as exception handling and workflow state managing. The implementation details of the execution coordinating service are demonstrated in Figure 3.

Figure 3 The implementation details of the execution coordinating service

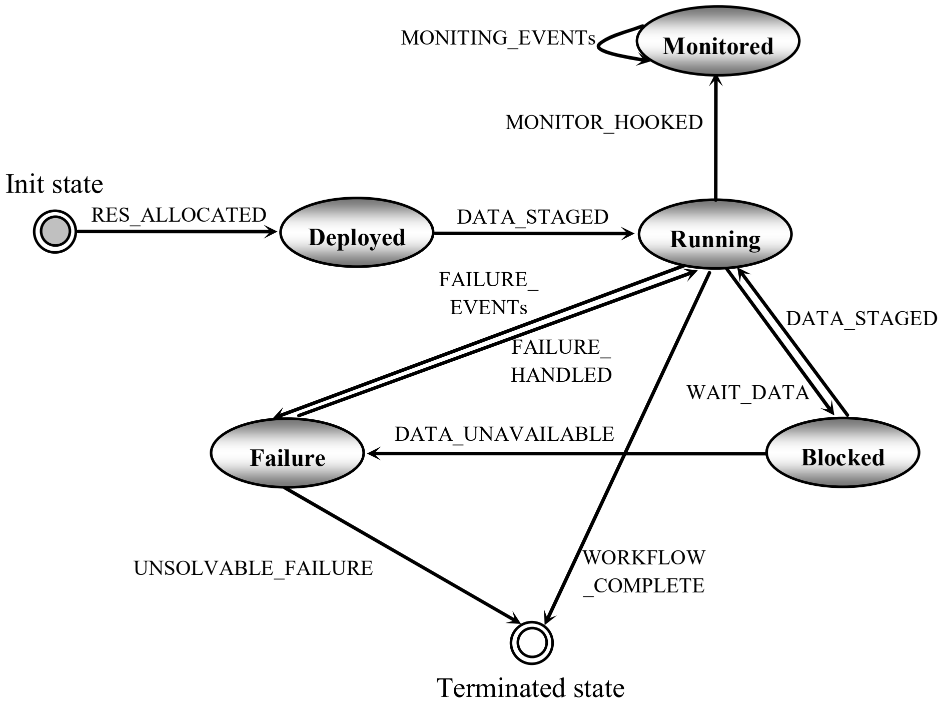


As depicted in Figure 3, the execution coordinating service mainly consists of four sub-components, including event service, performance monitor, task manager and state manager. As to the performance monitor and task manager, their functionalities are very similar to the conventional the components in the distributed resource management systems. The reason that we implement them in the execution coordinating service is to ensure that our proposed framework is self-contained and is able to be independently deployed in most of cloud platforms. In the following, we mainly focus on the implementations of the event service and state manager components.

The event service is for capturing the events that occur during the execution of workflows, which is designed based on the well-known subscription-notification model due to its scalability and extendibility. The other components in the coordinating service can subscribe their interesting events through event service, which is incorporated with a set of event channels for dispatching the pending events to the corrected subscribers. For instance, the performance monitor mainly wants to know the delivered SLA of virtualised resources, while task manager is mainly interesting in the availability of resources, execution efficiency, throughput, intermediate data, etc. Currently, we defined three types of events: task-related events, resource-related events and data-related events. The events can be easily added, deleted, or changed through a configuring file. Such an event-notification mechanism can ensure that our execution coordinating service is loosely coupled with the other cloud services (i.e., scheduler and VM manager).

With the help of event service, the state manager is introduced to obtain the overall status of any running workflows. Currently, the state manager in our CWOM uses the state transition model in Figure 4 to describe the runtime status of a workflow. In our CWOM framework, we introduce seven states to describe the status of a workflow, including *init*, *terminated*, *deployed*, *running*, *monitored*, *failure* and *blocked*. The transitions from one state to another are triggered by different events. Unlike those existing workflow state models, we define the deployed and monitored states with aiming at fine-grained management and controlling. The deployed state indicates that required resources have been allocated to the target workflow while the required dataset has not been staged yet. For some scientific workflows, this state might be very long as the required data might come from other workflows or services. So, the cloud resource brokers may resort to the advanced reservation mechanism to allocate resources or perform on-demand provisioning operations in an online manner. In some other scenarios where multiple workflows should be executed in a coordinated manner, the introduction of deployed state provides a better way to realise the goal of execution coordination. As to the monitored state, it is mainly designed for those workflows that require fine-grained monitoring service. It is a well-known fact that excessively using the monitoring service is likely to introduce extra overheads on either system resources or the application itself. So, when a workflow enters the monitored state, the cloud provider may enforce some resource optimisation policies to guarantee the efficiency and effectiveness of the target workflows. In addition, the monitored state can be used to describe the workflows which is been diagnosed by its runners. As to the failure and blocked states, they are mainly used to describe the conditions of resources and datasets which are used during the running of a workflow.

Figure 4 State transition model of running workflows



4 Experiments and performance evaluation

4.1 Experimental settings

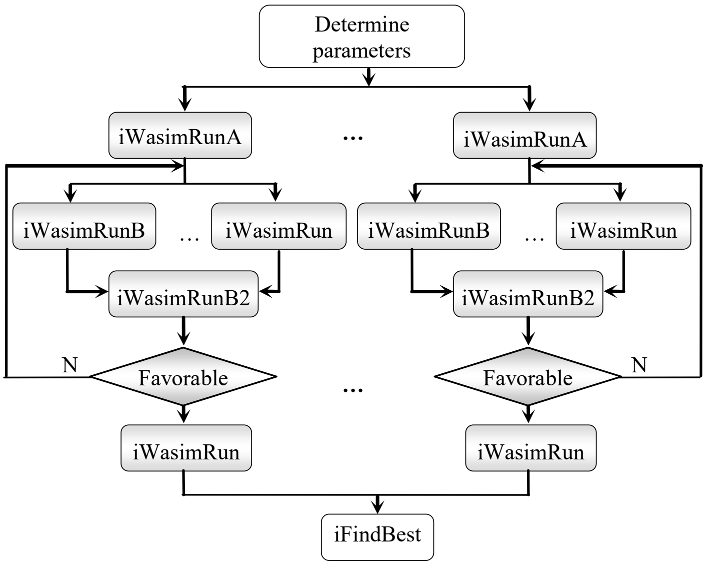
In this section, we present the experimental results obtained by deploying the CWOM system in a real-world cloud platform. The experimental test-bed cloud is constructed in our HP high-performance network centre, where the IT-infrastructures are consisted of twenty high-performance computing nodes (CN1~CN20) and seven massive storage nodes (SN1~SN7). All of these physical resources are virtualised by using XCP hypervisor, which provides a uniform VM pool and offers the basic executing hosts for the up-level applications. To evaluate the performance of CWOM, we select the well-known INVMOD workflow as the experimental workload, which is generally used to study the effects of climatic changes on the water balance. The basic model of INVMOD workflow is shown in Figure 5, which mainly consists of two parallel processing levels:

1 In the first level, the primary problem is divided into a set of similar sub-problems (iWasimRunA), each being solved by the same iterative algorithm.

2 In the second level, a parameter sweeping procedure is executed repeatedly until the desirable result is obtained.

At the end of INVMOD workflow, all the computing results (iWasimRun) are collected to get the final result. Each task in the two-level parallel processing will transfer its input data from storage nodes, and the intermediate data produced by iWasimRunB2 and iWasimRun needs be stored temporarily.

Figure 5 Framework of INVMOD application

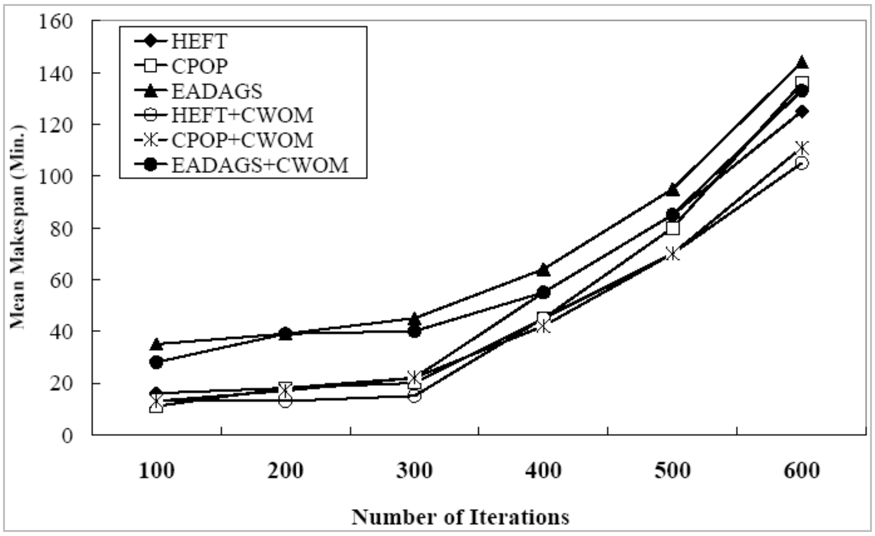


It is noteworthy that there exists a loop in the iterative algorithm template as shown in Figure 5. So, we set a pre-defined iteration counter (noted as *N*) to control the execution of INVMOD workflow. In this way, we cannot only adjust the intensiveness of resource requirements in our experiments, but also obtain different solutions of INVMOD with different accuracies. To investigate the interactions between CWOM and other workflow services (e.g., workflow scheduler and broker), we implement three widely used workflow scheduling and allocating algorithms, including HEFT (Topcuoglu et al., 2002), CPOP (Topcuoglu et al., 2002), and EADAGS (Lee and Zomaya, 2011). The HEFT and CPOP are two well-known heuristic scheduling algorithms with aiming at minimising the makespan of workflows, while the EADAGS algorithm is an energy-efficient workflow scheduling approach which relies on working-power states of resources to perform resource allocation and task scheduling operations. During our experiments, we first test the execution efficiency of INVMOD workflow without CWOM services, and then we turn on the services of CWOM and conduct the experiments with the same experimental conditions. In this way, we can evaluate the effectiveness of the proposed CWOM system.

4.2 Comparisons on makespan metric

In this section, we mainly take efforts on analysing the makespan metric, which is of significant importance to evaluate the execution efficiency of workflow applications. In our experiments, we gradually increase the iteration counter from 100 to 600 with aiming at adjusting the intensiveness of resource requirements. So, the parameter *N* can be considered as the intensiveness of system workloads. The experimental results are demonstrated in Figure 6.

Figure 6 Comparisons on makespan metric under different *N* values



At the first phase of our experiments, the CWOM is turn-off and the INVMOD workflow is executed in the conventional cloud platform. According to our experimental results shown in Figure 6, we can see that among the three tested schedulers, the makespan measurements obtained by using the EADAGS is the worst, while the performance of HEFT and CPOP is very close. This is because that the EADAGS always tries to sacrifice the execution performance for reducing the energy consumption of VM instances. When we turn on the CWOM system, we can see that the makespan measurements in all experimental are reduced regardless of the used schedulers. Specifically, when using the CWOM our CWOM can help it reduce the makespan measurements by about 3.5%~5.2% depending on the value of parameter *N*. When using the CPOP, the reduction of makespan metric is about 1.5%~6.4%. As to the EADAGS, the reduction of makespan metric is about –2.2%~7.8%.

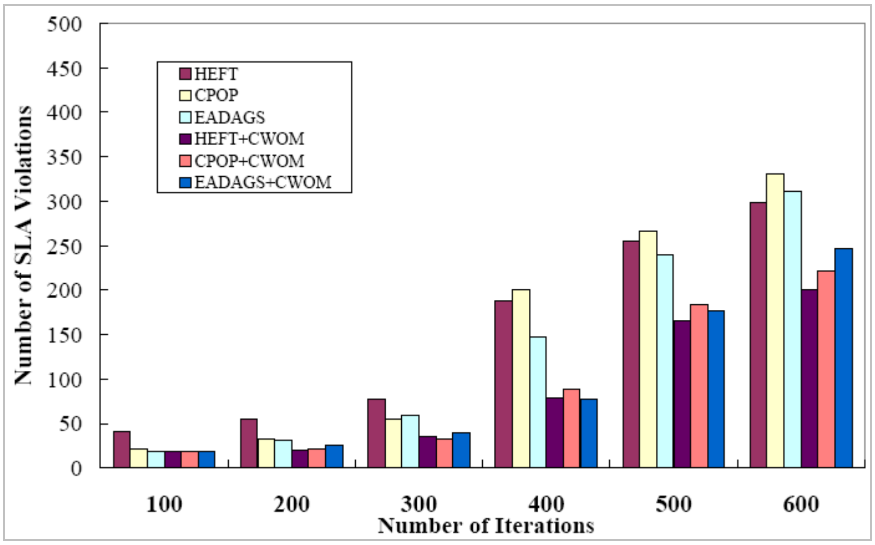
It is a well-known fact that the makespan metric of a running workflows may be affected by many factors, such as resource performance, workload intensiveness, resource management policy, security policy, resource availability, etc. Typically saying, it is difficult to say that which factor is the most important one since different workflow has various running characteristics. According to our experimental results, we can see that the number of iterations parameter directly decide the actual makespan metric. Even so, we still see that the makespan metric significantly increases only when the parameter *N* exceeds over 300. By analysing the experimental results, we find that the SLA matching service in CWOM system plays the most important role to reduce the makespan, which nearly contributes over 50% of the reduction of makespan metric in all experimental cases. So, we conclude that an effective SLA matching service can significantly improve the execution efficiency of workflow applications in cloud environments.

4.3 Comparisons on SLA violation metric

When deploying and running workflows, it is inevitable that SLA violations occur due to the different reasons. In this section, we focus on the occurred SLA violations in all experimental cases. Here, we only take into account three types of SLA violations, including resource failure, missing of task deadline, and data transferring delay. For the convenience of analysing, we add them together and use the sum of three kinds of SLA violation as the experimental metric. The experimental results are demonstrated in Figure 7.

According to results shown in Figure 7, we first notice that the number of SLA violations is very slightly when the number of iterations is below 200. In fact, such a violation level should be considered as the inherent runtime-feature of the test-bed platform. As long as the number of iterations increases, we can see that the frequency of violations grows rapidly. According to our experimental results, that the frequency of resource failure is maintained in a relative constant level, while the frequencies of missing task deadline and data transferring delay are more likely to be increased with increasing of parameter *N*. Based on the experimental results, it is clear that using the CWOM can significantly reduce the SLA violations. Specifically, the reduction of SLA violations obtained by CWOM is ranging between 25%~43%. As noted in Section 3, we know that the SLA matching service mainly focus on preventing the SLA violation before execution, while the execution coordinate service is responsible for deal with SLA violation during the execution of workflows. Therefore, both services play a positive role for reducing the SLA violations for deployed INVMOD workflow. Our experimental results also reveal that the tested schedulers seem to have a weak relation with the SLA violation metric. This observation may be useful for those cloud system, where the SLA performance is considered to be most important performance metric.

Figure 7 Comparisons on SLA violation metric under different *N* values (see online version for colours)

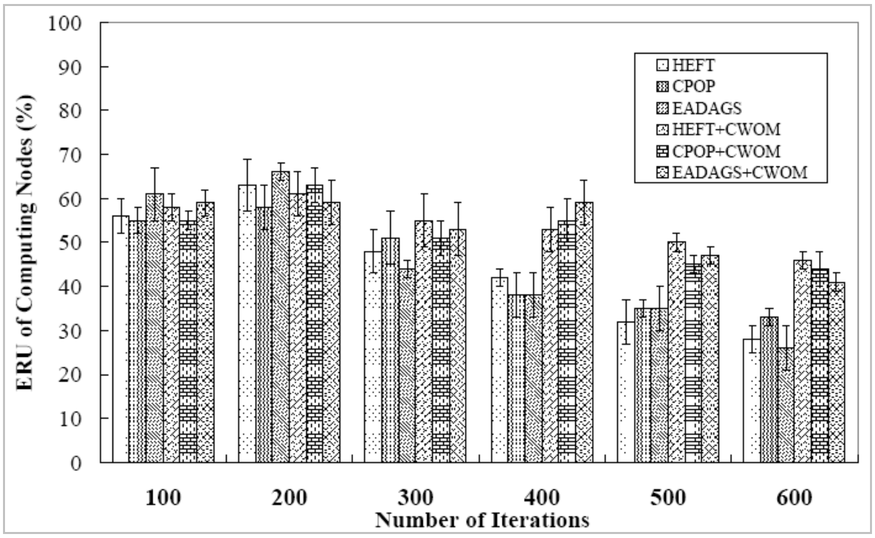


4.4 *Comparisons on resource utilisation metric*

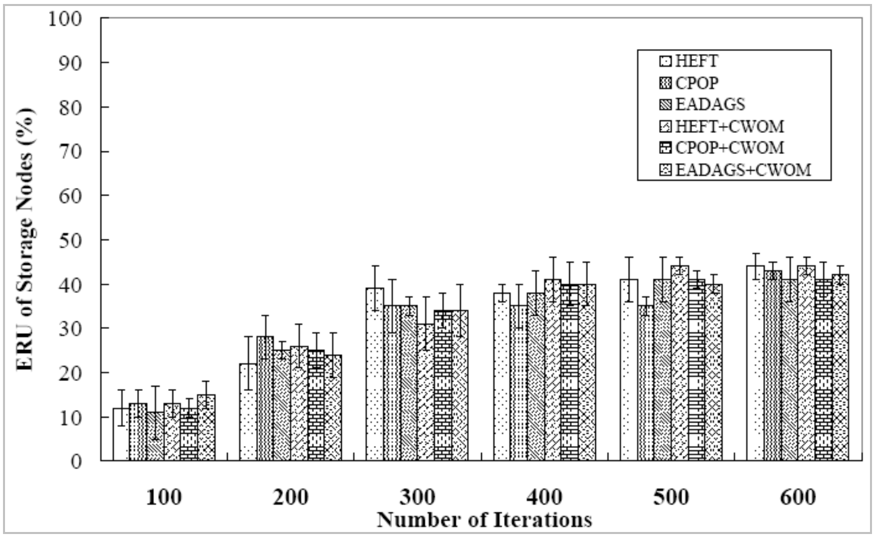
Lastly, we take efforts on analysing the resource metric in our experiments. Typically, this metric is used to evaluate that whether the underlying resources are fully utilised or not. Based on this metric, a cloud provider can adjust his resource provisioning policy to adapt to the system workloads. As our test-bed cloud platform offer the computing and storage capability through a common VM pool. Therefore, the conventional definition of resource utilisation metric may not be suitable since not all resources are virtualised at the same time. So, we introduce a new metric called effective resource utilisation (ERU), which only takes into account those resources used by VM hypervisor to form VM instances. As to the other physical computing and storage servers, we do not account them when calculating the ERU measurements. The experimental results are shown in Figures 8(a) and 8(b).

According to the results in Figure 8, we firstly can see that the EUR metrics of computing and storage nodes show different features. Specifically, the EUR metric of computing nodes is decreased with the increasing of parameter *N*, while the EUR metric of storage nodes is increase it. It is noteworthy that these results are decided by the feature of our tested workflow, which means that other kinds of workflows may show different trends. Even so, we can still tell that the proposed CWOM system can improve the EUR of computing nodes when the system is in presence of intensive workloads as shown in Figure 8(a). As to the storage nodes, we find that the CWOM system seems does not affect their ERU measurements significantly. This is because that the current implementation of our CWOM framework does not be incorporated with any data-related optimisation services. So, in the future, we are planning to design some data optimisation mechanisms in the CWOM framework so as to provide better data transferring or staging services for those data-intensive workflows.

Figure 8 Comparisons on ERU metric under different *N* values, (a) computing nodes (b) storage nodes



(a)



(b)

5 Summary and future work

In this study, we present a lightweight framework called CWOM, which is designed for easing the work of optimising the workflow execution efficiency as well as the work of realising fine-grained workflow management. The proposed CWOM is implemented as a plug-in middleware that can interact with the other workflow services and therefore provide more desirable services for workflow applications. In addition, the CWOM is loosely coupled with other cloud services, which makes it more flexible and configurable in most of the real-world cloud platforms. We conduct a set of experiments on a real-world cloud platform to evaluate its performance in terms of various performance metrics and the results show that the proposed CWOM system can significantly improve the delivered QoS and enhance the service level offered by conventional workflow solutions. In the future, we are planning to refine the implementation of CWOM and conduct more experiments on different cloud figurations, including different resource brokers and different IT-infrastructures. Also, we are planning to invent some data-related optimisation services and incorporate them into the CWOM framework with aiming to improve the execution efficiency of data-intensive workflows.

References

Abdi, S., PourKarimi, L. et al. (2018) 'Cost minimization for bag-of-tasks workflows in a federation of clouds', *Journal of Supercomputing*, Vol. 74, No. 6, pp.2801–2822.

Aslam, S., Islam, S.u. et al. (2017) 'Information collection centric techniques for cloud resource management: taxonomy, analysis and challenges', *Journal of Network and Computer Applications*, Vol. 100, No. 1, pp.80–94.

Bryk, P., Malawski, M. et al. (2016) 'Storage-aware algorithms for scheduling of workflow ensembles in clouds', *Journal of Grid Computing*, Vol. 14, No. 2, pp.359–378.

Byun, E-K., Kee, Y-S. et al. (2011) 'BTS: resource capacity estimate for time-targeted science workflows', *Journal of Parallel and Distributed Computing*, Vol. 71, No. 6, pp.848–862.

Cai, Z., Li, X. et al. (2017) 'A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds', *Future Generation Computer Systems*, Vol. 71, No. 1, pp.57–72.

Cao, H., Jin, H. et al. (2010) 'DAGMap: efficient and dependable scheduling of DAG workflow job in grid', *Journal of Supercomputing*, Vol. 51, No. 2, pp.201–223.

Casas, I., Taheri, J. et al. (2017) 'A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems', *Future Generation Computer Systems*, Vol. 74, No. 2, pp.168–178.

Deelman, E., Vahi, K. et al. (2015) 'Pegasus, a workflow management system for science automation', *Future Generation Computer Systems*, Vol. 46, No. 1, pp.17–35.

Delavar, A.G. and Aryan, Y. (2014) 'HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems', *Cluster Computing*, Vol. 17, No. 1, pp.129–137.

Diaz-Montes, J., Abdelbaky, M. et al. (2015) 'CometCloud: enabling software-defined federations for end-to-end application workflows', *IEEE Internet Computing*, Vol. 19, No. 1, pp.69–73.

Farkas, Z. and Kacsuk, P. (2011) 'P-GRADE portal: a generic workflow system to support user communities', *Future Generation Computer Systems*, Vol. 27, No. 5, pp.454–465.

Khaleel, M. and Zhu, M.M. (2016) 'Energy-efficient task scheduling and consolidation algorithm for workflow jobs in cloud', *International Journal of Computational Science and Engineering*, Vol. 13, No. 3, pp.268–284.

Krieger, M.T., Torreno, O. et al. (2017) 'Building an open source cloud environment with auto-scaling resources for executing bioinformatics and biomedical workflows', *Future Generation Computer Systems*, Vol. 67, No. 2, pp.329–340.

Le-Blanc, A., Brooke, J. et al. (2013) 'Workflows for heliophysics', *Journal of Grid Computing*, Vol. 11, No. 3, pp.481–503.

Lee, Y.C. and Zomaya, A.Y. (2011) 'Energy conscious scheduling for distributed computing systems under different operating conditions', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 8, pp.1374–1381.

Luo, H., Liu, J. et al. (2018) 'Predicting temporal violations for parallel business cloud workflows', *Software: Practice and Experience*, Vol. 48, No. 4, pp.775–795.

Marinho, A., Murta, L. et al. (2012) 'ProvManager: a provenance management system for scientific workflows', *Concurrency and Computation: Practice & Experience*, Vol. 24, No. 13, pp.1513–1530.

Moens, H. and De-Turck, F. (2015) 'Shared resource network-aware impact determination algorithms for service workflow deployment with partial cloud offloading', *Journal of Network and Computer Applications*, Vol. 49, No. 1, pp.99–111.

Qi, L. (2017) 'Workflow management system based on WEB technology', *Cluster Computing*, Vol. 20, No. 2, pp.941–947.

Rimal, B.P. and Maier, M. (2017) 'Workflow scheduling in multi-tenant cloud computing environments', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 28, No. 1, pp.290–304.

Rociszewski, P., Czarnul, P. et al. (2016) 'KernelHive: a new workflow-based framework for multilevel high performance computing using clusters and workstations with CPUs and GPUs', *Concurrency Computation*, Vol. 28, No. 9, pp.2586–2607.

Rodriguez, M.A. and Buyya, R. (2017) 'A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments', *Concurrency Computation*, Vol. 29, No. 8, pp.33–48.

Sookhak, M., Talebian, H. et al. (2014) 'A review on remote data auditing in single cloud server: taxonomy and open issues', *Journal of Network and Computer Applications*, Vol. 43, No. 1, pp.121–141.

Topcuoglu, H., Hariri, S. et al. (2002) 'Performance-effective and low-complexity task scheduling for heterogeneous computing', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 2, pp.260–274.

Wang, J., Huang, D. et al. (2017) 'SideIO: a side I/O system framework for hybrid scientific workflow', *Journal of Parallel and Distributed Computing*, Vol. 108, No. 1, pp.45–58.

Wang, Y., Lu, P. et al. (2015) 'WaFS: a workflow-aware file system for effective storage utilization in the cloud', *IEEE Transactions on Computers*, Vol. 64, No. 9, pp.2716–2729.

Weerasiri, D., Barukh, M.C. et al. (2017) 'A taxonomy and survey of cloud resource orchestration techniques', *ACM Computing Surveys*, Vol. 50, No. 2, pp.1–44.

Wu, F., Wu, Q. et al. (2015) 'Workflow scheduling in cloud: a survey', *Journal of Supercomputing*, Vol. 71, No. 9, pp.3373–3418.

Xiao, P. and Hao, Z. (2016) 'Improving energy-efficiency of large-scale workflows in heterogeneous systems', *International Journal of Computational Science and Engineering*, Vol. 13, No. 3, pp.258–267.

Yun, D., Wu, C.Q. et al. (2015) 'An integrated approach to workflow mapping and task scheduling for delay minimization in distributed environments', *Journal of Parallel and Distributed Computing*, Vol. 84, No. 1, pp.51–64.

Zeng, L., Veeravalli, B. et al. (2015) 'An integrated task computation and data management scheduling strategy for workflow applications in cloud environments', *Journal of Network and Computer Applications*, Vol. 50, No. 1, pp.39–48.

Zhao, Y., Li, Y. et al. (2015) 'Enabling scalable scientific workflow management in the cloud', *Future Generation Computer Systems*, Vol. 46, No. 1, pp.3–16.