J Supercomput (2014) 67:496–527
DOI 10.1007/s11227-013-1014-8

A grid workflow Quality-of-Service estimation based on resource availability prediction

Somayeh Kianpisheh · Nasrolah Moghadam Charkari

Published online: 25 October 2013

Abstract Accurate estimation of workflow Quality of Service (QoS) enhances the efficiency of scheduling algorithms. The availability and performance variations of Grid computing resources have made this estimation a great challenge. Most workflow QoS estimation algorithms are based on static performance of resources. In this paper, based on resources availability prediction, we propose an algorithm called WQE for estimating the QoS of a Grid workflow. WQE consists of two phases: resource monitoring and analysis and workflow QoS computation. In the first phase, two prediction algorithms are proposed to stochastically predict the availability state of resources. In the second phase, the QoS of each activity is estimated based on the host availability prediction result. The QoS of basic structures is computed by aggregating the QoS of their operands. Using a tree structure corresponding to the workflow, the QoS of basic structures is used to compute the total QoS of the workflow. The simulation results on Notre Dame University trace showed that the proposed method has higher estimation accuracy in comparison with HEFT.

Keywords Workflow · Grid computing · Quality of Service · Prediction · Multi-state system

1 Introduction

Distributed high performance computing systems, like grid and cloud computing, have provided infrastructure for resource sharing. Large scale grids like EGEE [1], TeraGrid [2] and PlanetLab [3] bring together sites of resources to facilitate e-science

S. Kianpisheh · N. Moghadam Charkari (✉)
Faculty of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran
e-mail: moghadam@modares.ac.ir

S. Kianpisheh
e-mail: s_kianpisheh@modares.ac.ir

and e-business issues. Grid middleware like Globus [4] and Condor [5] use scheduling algorithms to map jobs submitted by users to resources. Efficient job scheduling mechanisms are required to perfectly utilize the resources. An accurate job *Q*uality *o*f *S*ervice (QoS) prediction method enhances the efficiency of scheduling algorithms [6].

Job QoS estimation is a challenging problem due to the availability and performance variation of grid resources. For example, resources may be shut down or restarted to save power for a software update or even due to a hardware problem. During resource uptime, the performance may degrade because of a huge workload on the resource obliged by the owner of the resource or job grids. All these phenomena are obstacles to properly estimating the job QoS metrics.

Most of the current researches estimate the execution time of atomic jobs on resources [6–12]. However, many scientific and business processes are modeled via composite jobs or workflow. A workflow consists of several activities supposed to be executed according to a predefined order. Estimation of QoS parameters in a workflow is much more challenging than for an independent job since its execution involves multiple resources and activities. Two groups of researchers have considered the workflow QoS estimation: those working in the area of web service composition [13–16], and those in the area of workflow scheduling [17–20].

A composite web service invokes some web services to obtain the user requirements. The interactions among web services are modeled by a workflow. Studies have been done in the area of QoS estimation of composite web services [13–16] which cannot be applied for grid workflow for two reasons. First, in these works, the QoS estimation has been done independently of the mapping of activities to web services. In a grid, the resources are heterogeneous and different mappings of one workflow will have different QoS values. Second, these works have assumed a deterministic or semi-deterministic behavior for web services. For example, each web service has a fixed predefined response time or at most a predefined probability distribution function for response time. In a grid, resources are so dynamic that it is impractical to define a distribution or fixed quantity to model their response time, availability or cost.

During grid workflow scheduling, an estimation of QoS parameters (mostly makespan) is done to find a suitable mapping of activities to resources. Most of these algorithms like in [17–20] use the static built-in speed of resources to estimate the QoS parameters. The dynamicity in performance and availability of grid resources makes these estimations error prone.

In this paper, based on resources availability prediction, we propose a *W*orkflow *Q*oS *E*stimation algorithm called WQE. Figure 1 shows the input and output of the system. The workflow, the mapping of activities to resources[1] and the log of availability changes in resources are the input to the estimation system. The output is the QoS metrics consisting of response time, reliability and cost of workflow execution. Response time is the mean time in which the execution of the workflow completes. Reliability is the probability of the workflow execution completed without failure,

[1] The order of activities executions is also required when some parallel activities map to the same resource.

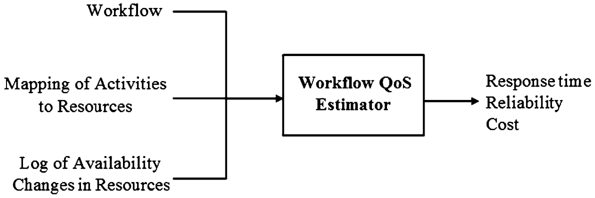

Workflow
Mapping of Activities to Resources
Log of Availability Changes in Resources
Workflow QoS Estimator
Response time
Reliability
Cost

Fig. 1 Input and output of a workflow QoS estimation system

and finally, cost is the budget user must pay for consuming resources. WQE consists of two main steps: resource monitoring and analysis and workflow QoS computation.

A multi-state availability model has been used for resources. The states define both the workload and availability of resources. In this paper, we use four availability states: Available, User present, CPU threshold exceeded, and Unavailable as used in the Grid failure trace archive [21]. A simple monitoring system monitors each resource and records the changes in the availability state within time. When a workflow is submitted, a window with a specified length is assumed on the availability trace of each resource. The availability information within this window is analyzed to predict the state of each resource. In this paper, two stochastic prediction methods are proposed: PE and PW. In PE, all historical information within the window has the same value while in PW; the information nearer to the prediction point gets more weight. The probability of being in a state is computed based on the percentage of time the state has occupied within the window. Simulation results indicate that PW performs better than PE in state prediction.

To compute the QoS of a workflow, the input workflow graph is converted to a tree structure. In the tree, leaf nodes are activities. Middle nodes are basic structures with their children from left to right indicating the operands they must be applied on. The QoS metrics are computed by aggregating the QoS of nodes, in a bottom-to-top manner according to the structure of the tree. To enable computation, at first the QoS of each activity must be computed. This computation is based on the prediction of the availability states of the host. Then, the QoS metrics of activities are aggregated to form the QoS of basic structures. At the end, these values are composed based on tree structure to generate the QoS of the workflow.

The method has been evaluated for random and real world workflows. We compare our work with Heterogeneous Earliest Finish Time (HEFT) QoS estimation [18]. To make this comparison, we get the output scheduling of HEFT as the input of our algorithm and estimate the QoS parameters. Then our estimation and HEFT estimation are compared with the actual values. These values are achieved by simulating workflow execution on the grid of Notre Dame University (NDU) according to Condor trace at 2007 [21]. For 25620 random workflows in simulation, in comparison with HEFT, WQE has gained 52 %, 41.3 % and 48.2 % improvement in reliability, response time and cost estimation, respectively.

In the rest of this paper, the related works are mentioned in Sect. 2. In Sect. 3, the problem statement and preliminaries are described. The details about our work will be discussed in Sect. 4. In Sect. 5, the complexity of the proposed algorithm will be analyzed. The results will be presented in Sect. 6, and finally, Sect. 7 gives the conclusion and directions for future work.

2 Related works

We categorize the related works into two groups: (i) resource state prediction, (ii) job QoS estimation. Each group is explained separately.

2.1 Resource state prediction

Some prediction methods for load on resources have been presented before. The Network Weather Service (NWS) uses a mixture of experts based on a linear model to choose the best prediction among experts [22]. Linear predictors have been proposed based on averaging the states during last N intervals [23]. Hu et al. proposed two methods of load prediction based on support vector regression and neural network. They emphasized using the least amount of features in prediction to reduce the cost of monitoring [24].

Byun et al. defined a Markov chain with three states for a resource: idle, in use and stopped. The resources are monitored every 30 minutes to estimate the rate of transitions among the states. The resource with the highest availability probability is employed for a submitted job [11]. Jiong et al. assumed two states, idle and busy, for each resource. With the assumption of static and predefined transition rates, they tried to boost job scheduling [25]. Lili and Shoubao specified CPU usage, network usage and failure of resources as Markov state variables [26]. They supposed a value for each state and aggregated these values to compute a rank for each resource. Resources with higher ranks were used during job scheduling. Ren et al. used CPU load, memory thrashing and unavailability of resources to form a Markov chain with five states. They emphasized on computing the probability of transitioning from available states to other states [27]. Rood and Lewis used the same states introduced in this paper but they performed a discrete transitional analysis to compute the probability of resource states [6]. They also took advantage of this analysis for independent job scheduling. Ramakrishnan and Reed suggested a birth-and-death process with six availability states to model resource state changes. Based on static transition rates, they predicted the probability of the resource being in each state [12].

2.2 Job QoS estimation

Most of the current researches predict the execution time of atomic jobs on resources. They perform this prediction directly [7–10] or based on resource state prediction [6, 11, 12]. Most workflow scheduling algorithms use static information about resources to estimate QoS of a workflow [17, 18, 20, 28–30]. In these works, the dynamicity of resources has been ignored and they do not support selection and loops in the structure of a workflow.

There are some works in QoS computation of composite web services. A composite web service is made of some tasks, each supposed to call a web service. The works in this area are categorized into two groups: In the first group, deterministic behavior for web services is assumed, while in the second group, the probability distributions of web service responses are considered to be well defined. Jaeger et al. estimated the QoS of a composite service by aggregating the QoS of individual

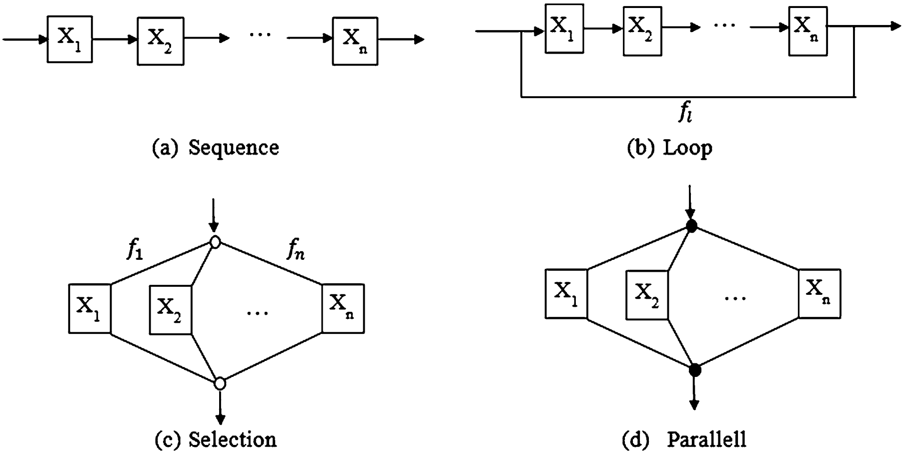


Fig. 2 Four basic structures used in workflows

services. They assumed a deterministic behavior of web services in responding to requests [15]. Cardellini et al. also estimated the QoS of a workflow with this assumption. They modeled service selection as a linear programming problem based on the computed QoS values. To adapt with dynamicity of web service behaviors, they proposed rescheduling. However, rescheduling makes overhead for a workflow management system [31]. Hwang et al. [14] used a probability mass function (PMS) for QoS of services. It aggregates the PMS to give a probabilistic QoS for a workflow. Zheng et al. introduced kernel estimation as a nonparametric method to estimate the probability density function of services. The PDFs were aggregated to generate the PDF of QoS for a workflow [16]. However, applying these methods on grid workflows is impractical because of resource heterogeneity and dynamicity.

3 Problem statement and preliminaries

In this section, we illustrate the formal definition for a workflow and grid resources, and describe the problem.

3.1 Workflow

A grid workflow can be represented by a directed graph $G = (A, E)$ where $A = \{a_1, a_2, \dots, a_M\}$ is a set of activities and E is the set of edges showing precedence relationships among activities. In a given workflow graph, the activity with no predecessor is an entry and the activity with no successor is called an exit activity. RE_i is the set of resources having software/hardware requirements needed for executing activity a_i. $\boldsymbol{D}$ is an $M \times M$ matrix where d_{ij} shows the amount of data transmitted from a_i to a_j.

A composition of four basic structures can be used in the workflow graph: sequence, loop, selection and parallel (Fig. 2). Each X_i can be regarded as an operand

Table 1 Symbols used in the proposed algorithm

Category	Symbol	Description
Workflow specification	A	Set of all activities with size M
	a_i	Activity i of the input workflow
	O_i	Number of operations in activity i
	RE_i	Set of resources capable of executing activity a_i
	f_i	Probability of selection for activity a_i
	f_l	Number of iterations in loop l
	$\boldsymbol{D}_{M\times M}$	Data transmission matrix among activities
Resource	RE	Set of all resources with size N
	r_j	Resource j
	AL	Availability levels defined for resources
	i_{al}	Index of availability level al[a]
	$q_{j,t}$	Availability state of resource j at time t
	CS_j	Computational speed of resource j in the highest availability level
	α_{al}	Resource speed degradation coefficient of availability level al
	$c_{i,j\|al}$	Cost of executing task i at resource j when it is in availability state al
	β_{cost}	Cost coefficient for executing one operation
Network	$\boldsymbol{BW}_{N\times N}$	Bandwidth matrix between resources
	$\boldsymbol{NL}_{1\times N}$	Network latency
Mapping policy	$\boldsymbol{M}_{M\times 1}$	Mapping policy vector
	$M(a_i)$	Index of resource responsible of executing activity a_i

[a] We define $\{0, 1, 2\}$ as the indices of available, user present, and CPU threshold exceeded, respectively. Note that in this paper, these indices are used for computing processing speed at each state. In NDU trace, different numbers have been used

for the correspondent basic structure. An operand can recursively involve a basic structure or be an atomic activity, i.e., $X_i \in \{seq, par, sel, loop\} \cup A$. In the loop structure, f_l is assumed to be the mean number of iterations, and in the selection block, f_i is the probability of selection of X_i. For sequence, loop and parallel structures, we have: $\forall X_i;\ \ f_i = 1$.

3.2 Grid computing resources

The set $RE = \{r_1, r_2, \dots, r_N\}$ is the set of all grid resources. The computational speed of resource r_j is represented by CS_j. $\boldsymbol{BW}$ is an $N \times N$ matrix where BW_{ij} indicates the bandwidth between resources r_i and r_j. $\boldsymbol{NL}$ is a vector of N elements where NL_j shows the network latency of resource r_j. A ready time of resource r_j is denoted by rt_j which is defined as the time the resource is waiting to accept a new process after it has finished processing previously assigned processes. Table 1 shows the summary of symbols used in the proposed method.

```
HEFT Algorithm
Input
    G(A,E) : A graph representing the workflow
    RE: set of all resources
Output
    M: The workflow schedule
    QoS: $\{\mu\text{: makespan}, r\text{: reliability}, c\text{: cost}\}$
Procedure
Compute the rank value for each activity
    $rank(a_i) = w_i \quad if\ a_i\ is\ leaf$
    $rank(a_i) = w_i + \max_{a_j \epsilon succ(a_i)} (w_{i,j} + rank(a_j))$
where $w_i$ is the average execution time for each activity and $w_{i,j}$ is the average data transfer time
between each activity $a_i$ and its successor $a_j$
    $w_i = \frac{\sum_{i \epsilon RE_i} \frac{O_i}{CS_j}}{|RE_i|}$        $w_{i,j} = \frac{\sum_{r_i \epsilon RE_i, r_j \epsilon RE_j} \frac{d_{i,j}}{BW_{i,j}}}{|RE_i| \times |RE_j|}$
Sort the activities in a scheduling list $Q$ in decreasing order of their rank.
While $Q$ is not empty do
    $a \leftarrow$ remove the first activity from $Q$
    $r \leftarrow arg\, min_j\, EST_{a,j} + \frac{O_i}{CS_j}$   // $EST_{a,j}$ is the earliest start time of activity $a$ on resource $r_j$
    Schedule $a$ to $r$ ($M_a \leftarrow r$) and update ready time of resource $r$
End While
$\mu \leftarrow$ makespan, $r \leftarrow 1$, $c \leftarrow$ cost
```

Fig. 3 HEFT algorithm

3.3 Problem statement

Let $G = (A,\ E)$ be the workflow and $RE = \{r_1, r_2, \ldots, r_N\}$ be the set of resources. Let $\boldsymbol{M}$ be an $M \times 1$ vector where $M(a_i)$ shows the index of the host for activity a_i. The problem is to estimate QoS parameters for G under the mapping M, relative to the dynamic behavior of RE. The QoS parameters include: *Response time* which is the mean time where the execution of the workflow completes; *Reliability* which is the probability of the workflow execution being completed without failure; *Cost* which is the budget user should pay for consuming resources.

3.4 HEFT algorithm

Figure 3 shows the HEFT algorithm [18]. Accordingly, activities and edges are assigned some weights. The weight of each activity defines its average execution time on resources. The weight of each edge is the average required time to transfer data from the predecessor to the successor activity. By traversing the graph of the workflow from the bottom to the top, and using a recursive relation, a rank is computed for each activity. The ranks of leaf activities, i.e., activities without any successor, are the same as their weight. The ranks of other activities are the maxima of sums of ranks of their immediate successors and the weights of connecting edges. This ranking gives execution priority to critical activities to minimize the makespan. Activities are sorted by these ranks to form a list. They are removed from this list one by one, and each activity is assigned to a resource with minimum completion time. In this algorithm, resources are assumed to be available and have static computational speed. The mapping of activities to resources and the estimated values for QoS parameters including response time (makespan), reliability and cost are regarded as the output of this algorithm. We use HEFT mapping as the input to our proposed method and compare our estimation of QoS with HEFT estimation.

The HEFT does not support selection and loop structure. A loop structure can be considered as the repetition of the structure for a specific number of iterations. For selection some solutions are possible:

- For every workflow that consists of selection, construct a set of workflows. Each workflow in this set is a possible execution of the original workflow with its probability of execution computable according to the selection probabilities. For example, for a workflow which has two selection structures with selection probabilities (0.2, 0.8) and (0.4, 0.3, 0.3), six workflows with probabilities 0.08, 0.06, 0.06, 0.32, 0.24, 0.24 will be constructed. For each workflow a mapping is generated by HEFT. The QoS parameters are computed as weighted sums of the QoS of each workflow where the weights are execution probabilities of workflows.
- The selection structure can be substituted by a single activity with its number of operations obtainable as a weighted sum of the number of operations of tasks involved in the structure. The selection probabilities form the weights.
- Ignoring the selection probabilities, a mapping can be found for the workflow. Regarding the found mapping, the QoS parameters are recomputed as the weighted sums of QoS of the activities where the weight of activity a_i is equal to f_i.

In this paper, we used the third solution because it does not have the computation overhead of the first solution and it does not omit the selection structures as in the second solution.

4 Workflow QoS estimation system

Figure 4 shows the architecture of the proposed method. There are two main components:

(a) Resource monitoring and analysis—To detect the dynamic behavior of resources, we employ a simple monitoring system. Some availability states are defined for resources. These states represent load of works on resources. A higher availability state implies a lower workload on the resource, which leads to a better processing capability. The monitoring system monitors each resource periodically and logs the changes in the availability states. A stochastic predictor analyzes the log to predict the availability state of each resource.

(b) Workflow QoS computation—In this part, the QoS of a workflow is computed based on the resource analysis result. To achieve this goal, the QoS of each activity is estimated after data transmission modeling. Then, the quality of service for basic structures involved in the workflow is computed. Finally, the QoS composition estimates the whole workflow QoS by traversing a tree corresponding to the workflow from the bottom to the top.

In the rest of the paper, we explain each component in detail.

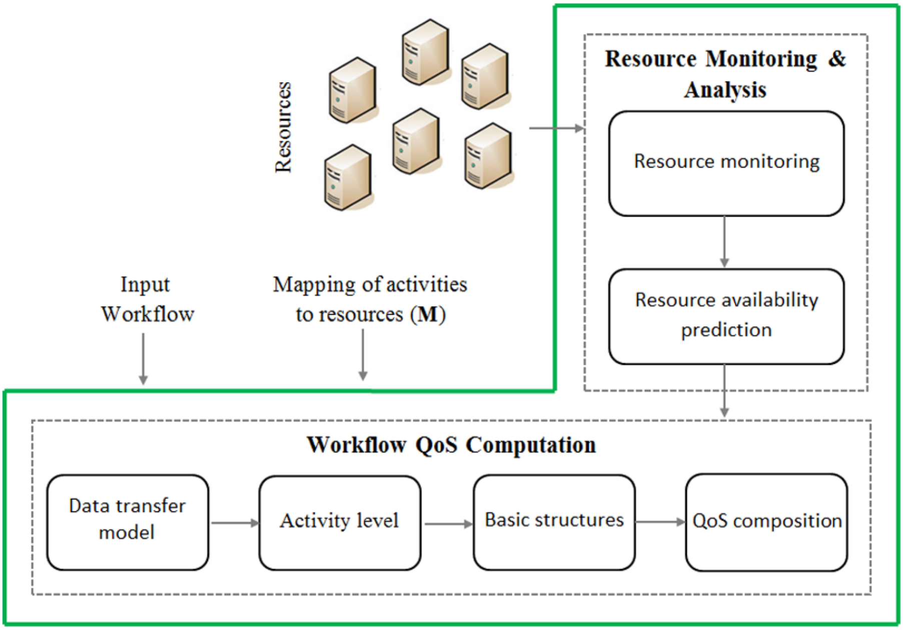


Fig. 4 Architecture of the proposed method for workflow QoS estimation

4.1 Resource monitoring & analysis

A simple monitoring system monitors each resource periodically to record the changes in availability state of the resource with time. In this paper, we use the four availability states supported by Condor [5, 6]: *available*, *user present*, *CPU threshold exceeded*, and *unavailable*. An *available* machine is currently connected to the network, has more than 15 minutes of idle time, and a CPU load less than the CPU threshold.[2] In the *user present* state, the resource owner has touched the keyboard or mouse. In *CPU threshold exceeded*, the local CPU load surpasses some specific threshold, due to new or currently running processes. Finally, when a machine fails or becomes unreachable, it will be *unavailable*.

The multi-state system assumed for modeling the changes in the availability of a resource is shown in Fig. 5. Rood et al. proposed a discrete transitional availability state predictor for atomic jobs [6], which is not suitable for our work. As a workflow has many activities, it takes a long time to be processed. In this regard, the Rood's predictor misses most of the information in the history, which leads to low information for prediction. So, other predictors must be employed.

Figure 6 shows the concept of our predictor. Assume we want to predict the state of a resource at time t. A window with size L is used, which contains the latest historical information during time interval $[t - L, t - 1]$. Inside the window, there are

[2] Generally, the CPU threshold value can be defined by resource owners. The Condor uses 30 % as CPU threshold for an available state [6].

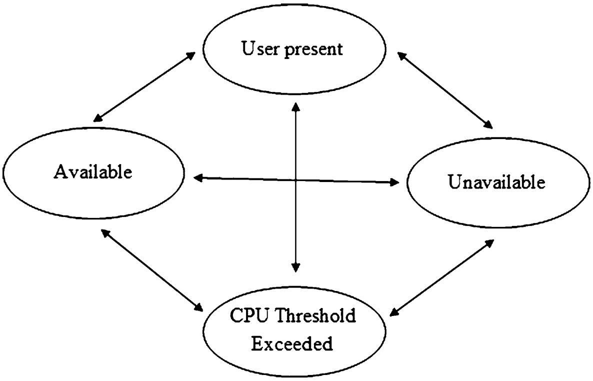


Fig. 5 State diagram used for modeling the behavior of resources in the NDU Trace

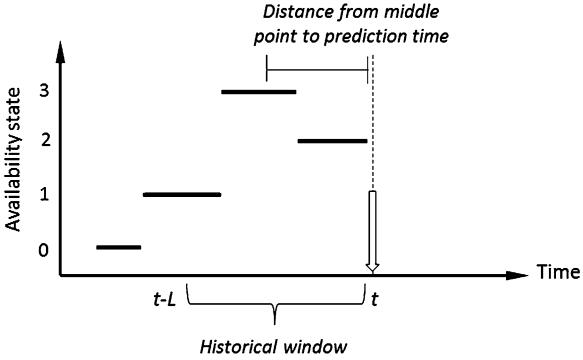


Fig. 6 Concept of resource state prediction using historical information inside a window with size L

some subintervals, each corresponding to a specific availability state. For instance, as it is shown in Fig. 6, there are three subintervals with availability states 1, 3, and 2, respectively. We define the distance from the middle of subinterval k (in the time range of $[t_1^k, t_2^k]$) to time t as in Eq. (1):

$$d_{mt}^k = t - \frac{t_1^k + t_2^k}{2} \tag{1}$$

Two predictors are proposed: *P*rediction with *E*qual Weights (PE) and *P*rediction with *W*eighting (PW). In PE, all observations have the same weight while in PW the observations nearer to the prediction time are regarded to be more important, hence they will get more weight. The reason for observation weighting is that our investigation on the NDU trace [21] has shown a behavior similar to the Markov property in state changes of resources. In other words, a future state of a resource depends mostly on the sequence of earlier events than the later ones. So PW tries to give more weight to observations closer to the prediction point, in order to enhance the prediction accuracy. The weight of an observation for resource j at time i belonging to subinterval s is defined in Eq. (2):

$$w_{j,i} = \begin{cases} 1 & \text{PE} \\ \frac{\min_k d_{mt}^k}{d_{mt}^s} & \text{PW} \end{cases} \tag{2}$$

```
Resource State Predictor
Input
    $L$: size of window
    Log file: resources historical availability trace
    $t$: time of prediction
Output
    $\{\pi_{j,al}(t) | \forall r_j \in RE, \forall al \epsilon AL\}$: Probability of resource j being in availability state al at time t
Procedure
    For each resource $r_j$ do
        From availability log file, get observations of resource $r_j$, from time t-L to t-1.
        For each availability state al
            If predictor is PW
                $SI_{al} = \{k | k\ is\ a\ subinterval\ within\ window\ with\ observations\ al\}$
                For each $k \in SI_{al}$
                    $t_1^k \leftarrow$ start time of observations in subinterval k
                    $t_2^k \leftarrow$ end time of observations in subinterval k
                    Use (1) to compute $d_{mt}^k$
                End-For
                $k' \leftarrow arg\ min_{k \in SI_{al}} d_{mt}^k$
                For each $k \in SI_{al}$
                    Set the weight of any observation at time $i \in [t_1^k, t_2^k]$ as:
                    $w_{j,i} = \frac{d_{mt}^{k'}}{d_{mt}^k}$
                End-For
            End-IF
            If predictor is PE
                For any observation at time $i \in [t - L, t - 1]$: $w_{j,i} = 1$
            End-IF
            Use (3) to compute $\pi_{j,al}(t)$
        End-For
    End-For
```

Fig. 7 Stochastic resource availability state predictor

In PE, all observations get 1 as the weight. In PW, the weight is computed in such a way that the observations belonging to the nearest subinterval to the prediction time, have weight equal to 1. Every other observation has weight reversely proportional to the distance of the middle point of its subinterval to the prediction time. In Eq. (2), k is an arbitrary subinterval within the window. With this weighting, all observations within one subinterval get equal weight which is reversely proportional to their average distance from prediction point. The probability that resource j is at state al at time t is computed by the percentage of time the resource was at al according to the history in the window as defined by Eq. (3):

$$\pi_{j,al}(t) = \frac{\sum_{i=t-L}^{t-1} w_{j,i}(q_{j,i} = al)}{\sum_{i \in [t-L,t-1]} w_{j,i}} \tag{3}$$

where $q_{j,i}$ is the observation for resource j at time i and $(q_{j,i} = al)$ is equal to 1 where the state of resource j at time i is al otherwise it is zero. Figure 7 shows steps of resource state prediction.

4.2 Workflow QoS computation

To compute the QoS of a workflow under the mapping $\boldsymbol{M}$, the QoS is estimated at the level of activity. These estimates are aggregated to form the QoS of the basic structures. The composition of the QoS of the basic structures results in the QoS of the whole workflow. In the following sections, each part is explained separately.

4.2.1 Data transfer time modeling

Before the execution of each activity, its required data must be transmitted from predecessors host to the activity host. We assume that the data transfer time between any two activities has a normal distribution with the mean value computed by Eq. (5). It is the relation of the size of data transmitted between two activities to the bandwidth of hosts. The network latency has also been considered. The variance of data transmission can be determined according to network communication.

$$\forall a_i, \quad a_j\, dt\, t(a_i, a_j, \boldsymbol{M}) \sim N\big(\mu\, dt(a_i, a_j, \boldsymbol{M}), \sigma dt^2(a_i, a_j, \boldsymbol{M})\big) \tag{4}$$

$$\mu\, dt(a_i, a_j, \boldsymbol{M}) = NL_{M(a_i)} + \frac{d_{ij}}{BW_{M(a_i),M(a_j)}} \tag{5}$$

4.2.2 Activity level

In this section, we estimate the QoS parameters of an arbitrary activity a_i. Before the start of execution of an activity, some latency (possibly zero) will exist. This latency is due to the time needed for transmitting data from predecessors to the activity or due to the current process on the mapped resource. Equation (8) shows the computation of the latency. Let FT_{a_j} be the finish time of activity a_j and $rt_{M(a_i)}$ be the ready time of a_i's host. The finish time of data transmission is computed by Eq. (6), and the latest predecessor is defined by Eq. (7). The first equation in (8) occurs when the data has been transmitted but the host is not ready to process the activity. In this case, the difference of the ready time and finish time of the latest predecessor defines the latency. The second equation in (8) occurs when the resource is waiting for completion of data transmission. In this case, the latency is equal to data transmission time from the latest predecessor.

$$FTDT = \max_{a_j \in \mathrm{pred}(a_i)} \big\{FT_{a_j} + \mu\, dt(a_j, a_i, \boldsymbol{M})\big\} \tag{6}$$

$$a_p = \operatorname*{argmax}_{a_j \in \mathrm{pred}(a_i)} \big\{FT_{a_j} + \mu\, dt(a_j, a_i, \boldsymbol{M})\big\} \tag{7}$$

$$\mathrm{lat}(a_i, \boldsymbol{M}) = \begin{cases} rt_{M(a_i)} - FT_{a_p} & rt_{M(a_i)} > FTDT \\ \mu\, dt(a_p, a_i, \boldsymbol{M}) & rt_{M(a_i)} \le FTDT \end{cases} \tag{8}$$

Let CS_j be the built-in computational speed of resource j. We assume that computational speed reduces by a coefficient of α_{al} in the availability state al as computed by Eq. (9). The degradation coefficient is specified by statistic information.

$$CS_{j|al} = \begin{cases} CS_j(1-\alpha_{al}) & al \in AL \backslash \text{unavailable} \\ 0 & al: \text{unavailable} \end{cases} \tag{9}$$

The expected time of executing activity a_i when the host is in state al is computed by Eq. (10) where O_i is the number of operations in a_i.

$$t_{i,M(a_i)|al} = \frac{O_i}{CS_{M(a_i)|al}} \tag{10}$$

Using Eqs. (8), (9), (10) and the prediction result gained from predictor, the mean response time is computed by Eq. (11) which is latency plus a weighted sum of the probability vector of resource states and expected execution time of activity in each state. Note that the mean response time of an activity shows the required time the resource successfully completes the activity execution. So, in the computation, unavailable states have been omitted, and the probability of each state under the condition of the resource being up has been used.

$$\mu(a_i, \boldsymbol{M}) = \text{lat}(a_i, \boldsymbol{M}) + \sum_{al \in AL \backslash \text{unavailable}} \frac{\pi_{M(a_i),al}}{\sum_{al \in AL \backslash \text{unavailable}} \pi_{M(a_i),al}} \times t_{i,M(a_i)|al} \tag{11}$$

The probability of a successful execution of an activity is computed by multiplying two probabilities: the probability that required data has successfully been transmitted and the probability that a resource is available. This probability is found as in Eq. (12):

$$r(a_i, \boldsymbol{M}) = \prod_{a_p \in \text{pred}(a_i)} r\,dt(a_p, a_i, \boldsymbol{M}) \times \sum_{al \in AL \backslash \text{unavailable}} \pi_{M(a_i),al} \tag{12}$$

The execution cost of activity a_i when the host is in state al is computed by Eq. (13). This cost has a direct relation to the number of operations of the activity and the computational speed of the host at availability state al. In this equation, the division by the maximal computational speed is for normalization. β_{cost} is a cost coefficient which is assumed for execution of each operation. This coefficient is specified by the grid owners according to their economic policy.

$$c_{i,M(a_i),|al} = O_i \times \beta_{\text{cost}} \times \frac{CS_{M(a_i)|al}}{\max_{j \in RE_i} CS_j} \tag{13}$$

The total execution cost of activity a_i is computed by a weighted sum of the probability vector of resource states and the expected cost of execution in each state as in Eq. (14):

$$c(a_i, \boldsymbol{M}) = \sum_{al \in AL \backslash \text{unavailable}} \frac{\pi_{M(a_i),al}}{\sum_{al \in AL \backslash \text{unavailable}} \pi_{M(a_i),al}} \times c_{i,M(a_i)|al} \tag{14}$$

Table 2 QoS of basic structures. $\boldsymbol{X_i} \in \{seq, par, sel, loop\} \cup A$

Basic structure	Quality of Service	Basic structure	Quality of Service
Sequential	$\mu(seq, \boldsymbol{M}) = \sum_{i=1}^{n} \mu(X_i, \boldsymbol{M})$	Parallel	$\mu(par, \boldsymbol{M}) = \max_{i=1,\ldots,n} \mu(X_i, \boldsymbol{M})$
	$r(seq, \boldsymbol{M}) = \prod_{i=1}^{n} r(X_i, \boldsymbol{M})$		$r(par, \boldsymbol{M}) = \prod_{i=1}^{n} r(X_i, \boldsymbol{M})$
	$c(seq, \boldsymbol{M}) = \sum_{i=1}^{n} c(X_i, \boldsymbol{M})$		$c(par, \boldsymbol{M}) = \sum_{i=1}^{n} c(X_i, \boldsymbol{M})$
Selection	$\mu(sel, \boldsymbol{M}) = \sum_{i=1}^{n} f_i \times \mu(X_i, \boldsymbol{M})$	Loop	$\mu(loop, \boldsymbol{M}) = f_l \times \mu(seq, \boldsymbol{M})$
	$r(sel, \boldsymbol{M}) = \sum_{i=1}^{n} f_i \times r(X_i, \boldsymbol{M})$		$r(loop, \boldsymbol{M}) = r(seq, \boldsymbol{M})^{f_l}$
	$c(sel, \boldsymbol{M}) = \sum_{i=1}^{n} f_i \times c(X_i, \boldsymbol{M})$		$c(loop, \boldsymbol{M}) = f_l \times c(seq, \boldsymbol{M})$

4.2.3 QoS of basic structures

The QoS of basic structures are computed as shown in Table 2. In a sequence structure, the response times of all operands are accumulated and the reliabilities of them are multiplied. In a parallel structure, the maximal response time of all operands forms the final response time and the reliability is again found by multiplication of reliabilities of all operands. The cost in these structures is the sum of costs for all operands. The computation is similar for selection. The only difference is that the probabilities of selection of operands are involved in the computation. A loop can be considered as a sequence structure which repeats for iterations.

4.2.4 QoS composition

The input of the system is a directed graph representing the workflow. This graph is converted to a tree structure (T). In the tree, the leaf nodes are activities and the middle nodes are basic structures with their children from left to right indicating the operands the basic structure must be applied on. An example is shown in Fig. 8. The tree structure can be constructed by syntax definition of the workflow. To compose the whole QoS, it is enough to call $\{\mu(root(T), \boldsymbol{M}), r(root(T), \boldsymbol{M}), c(root(T), \boldsymbol{M})\}$. Figure 9 shows the sequence of calls occurred during the mean response time computation from the tree structure in Fig. 8. The whole algorithm based on the mentioned steps has been shown in Figs. 10 and 11.

4.3 An illustrative example

To demonstrate the steps of the algorithm, in this section we illustrate by an example. Figure 12 shows a workflow to be executed on three resources. The labels on the edges show the data amount transmitted among activities. For simplicity, the inter-bandwidth among all resources is assumed to be 1 while the intra-bandwidth is ∞ and the data transmission variance is zero. The resources change their behavior as shown in Fig. 13. The fastest resource r_1 starts with a user present and transitions to *CPU threshold exceeded* at time 50. It becomes *unavailable* at time 200. The slowest resource r_3 is permanently *available*. The mediocre resource r_2 is in *CPU threshold*

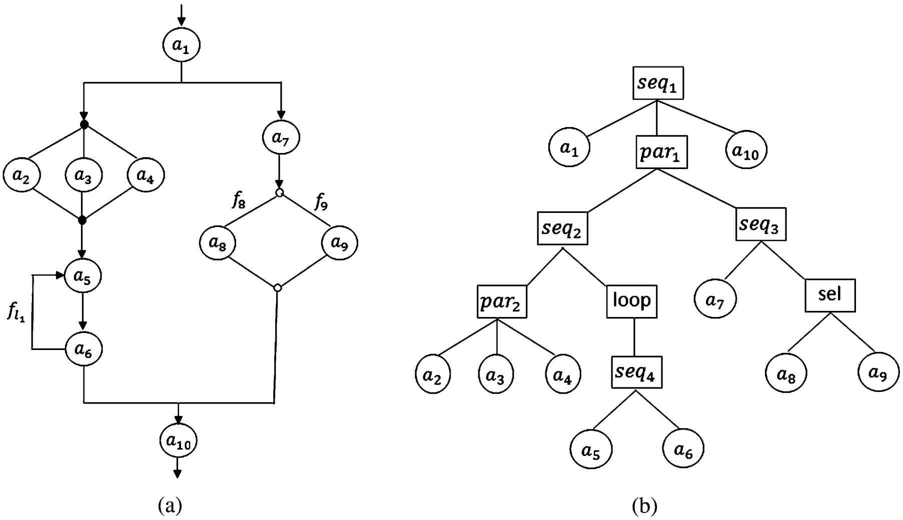
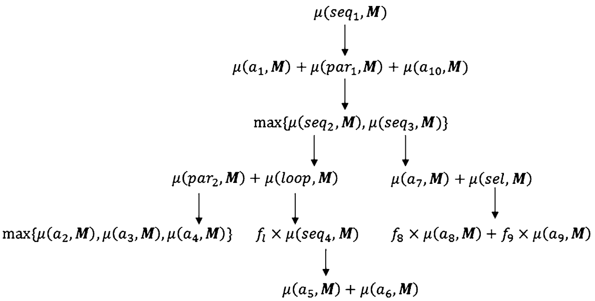


Fig. 8 An example workflow and its corresponding tree structure

Fig. 9 Sequence of calls for mean response time computation of sample tree structure in Fig. 8


$\mu(seq_1,\boldsymbol{M})$
$\mu(a_1,\boldsymbol{M}) + \mu(par_1,\boldsymbol{M}) + \mu(a_{10},\boldsymbol{M})$
$\max\{\mu(seq_2,\boldsymbol{M}), \mu(seq_3,\boldsymbol{M})\}$
$\mu(par_2,\boldsymbol{M}) + \mu(loop,\boldsymbol{M})$
$\mu(a_7,\boldsymbol{M}) + \mu(sel,\boldsymbol{M})$
$\max\{\mu(a_2,\boldsymbol{M}), \mu(a_3,\boldsymbol{M}), \mu(a_4,\boldsymbol{M})\}$
$f_l \times \mu(seq_4,\boldsymbol{M})$
$f_8 \times \mu(a_8,\boldsymbol{M}) + f_9 \times \mu(a_9,\boldsymbol{M})$
$\mu(a_5,\boldsymbol{M}) + \mu(a_6,\boldsymbol{M})$

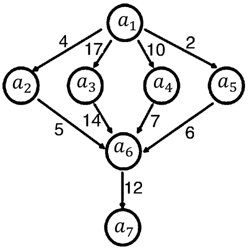
Workflow QoS Estimation Algorithm

```
Input
G: workflow graph
t: workflow submission time
M: Mapping of activities to resources
Output
Expected workflow QoS includes response time, reliability and cost
Procedure
1. Call resource predictor (PE or PW) to get the stochastic prediction of resources
   availability state at time $t$ : $\boldsymbol{\pi}$: $\{\pi_{j,al}(t)|\forall r_j \in RE, \forall al \epsilon AL\}$ // Fig. 7
2. Convert $G$ to a tree structure $T$
3. Return Node-QoS(root(T), $\boldsymbol{\pi}, \boldsymbol{M}$)
```

Fig. 10 The pseudocode of workflow QoS estimation

```
Node-QoS
Input
$node$: Node of tree
$\boldsymbol{\pi}$: $\{\pi_{j,al}(t) | \forall r_j \in RE, \forall al \epsilon AL\}$
M: Mapping of activities to resources
Output
qos: Expected workflow QoS includes response time, reliability and cost
Procedure
If $node$ is an activity use Eq. (11), (12), (14) to compute QoS parameters:
$qos \leftarrow \{\mu(node, \boldsymbol{M}), r(node, \boldsymbol{M}), c(node, \boldsymbol{M})\}$
End-If
If $node$ is a basic structure     // it is not a leaf in the tree
    $X = \{X_i | X_i \text{ is a child of node}\}$
    $n \leftarrow$ number of children
    For each $X_i \in X$
        $\boldsymbol{qos_i} \leftarrow$ Node-QoS($X_i$, $\boldsymbol{\pi}, \boldsymbol{M}$)
    End-For
    $qos \leftarrow$Aggregate $\{qos_i | i = 1..n\}$ based on its structure {"sequential",
    "parallel", "selection", "loop"}  according to Table 2.
End-If
```

Fig. 11 The pseudocode for computing the QoS of a node in tree



activity	O_i	Rank
a_1	40	81.3
a_2	70	52.4
a_3	50	58.4
a_4	100	58.8
a_5	80	54.9
a_6	110	37.0
a_7	60	8.8

$\forall al \in AL \quad \alpha_{al} = i_{al} * 0.3$

Resource	CS_i (op/sec)
r_1	10
r_2	7
r_3	5

$$BW = \begin{bmatrix} \infty & 1 & 1 \\ 1 & \infty & 1 \\ 1 & 1 & \infty \end{bmatrix}$$

$$NL = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

(a) Workflow structure (b) Activities information (c) Resource speeds (d) Network property

Fig. 12 An example workflow supposed to be executed on three resources. Ranks in part (**b**) are computed by HEFT as described in Fig. 3

exceeded state. In this example $\beta_{\text{cost}} = 0.01$ and $\forall al\ \alpha_{al} = 0.3$. The predictor is PW with window size of 350 s.

The workflow is submitted at times 70 and 250. The HEFT and WQE estimations of workflow execution at submit time 70 have been shown at Fig. 14. The actual execution of workflow has also been shown for comparison. As it is found, the WQE is much closer to the actual execution than HEFT. Note that the actual execution

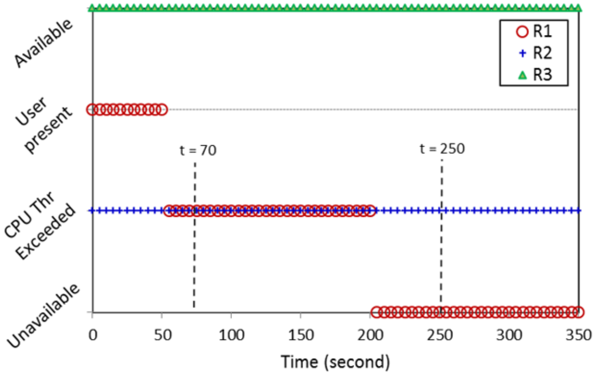


Fig. 13 Resources behavior

can be traced according to resources behavior in Fig. 13. For example, at time 70 the r_1 is in *CPU threshold exceeded* and has a speed of $CS_{1|\text{CPU thr execeeded}} = 10 \times (1 - 2 \times 0.3) = 4$ ops/s according to Eq. (9). Thus, submitting a_1 at time 70 takes $\frac{O_1}{CS_{1|\text{CPU thr execeeded}}} = \frac{40}{4} = 10$ s.

The final results of the response time, reliability and cost estimation in WQE, HEFT and the actual values are shown in Table 3. At time 250, the r_1 is *unavailable* and the workflow execution fails. The HEFT assumes the execution is reliable while the WQE computes the reliability as 0.013. For more clarification, some parts of the computations have been shown in Table 3.

5 Complexity analysis

In this section, we analyze the time complexity of the proposed method for estimating QoS metrics for a workflow with M activities and E edges supposed to be mapped on N resources defined in the mapping vector $\boldsymbol{M}$. Before proceeding with the analysis, the following theorems are proved.

Theorem 5.1 *The number of edges in a structured workflow with M activities as defined in Sect. 3.1 is $O(M)$.*

Proof Each activity is inserted to a workflow through a basic structure including sequence, parallel, selection, and loop. As it is observable in Fig. 2, this insertion to any arbitrary basic structure adds at most two more edges to the workflow. Thus, the number of edges is at most twice of the number of activities, i.e., $E = O(M)$. □

Theorem 5.2 *The number of nodes in the correspondent tree of any structured workflow with M activities is $O(M)$.*

Proof The minimum possible corresponding tree has two levels with one root and M activities as leaves. This tree obviously has $O(M)$ nodes. As the number of leaves is constant and equal to M, to construct a tree with maximum possible nodes, the middle nodes should be maximized. Ignoring loops, these nodes can be labeled as

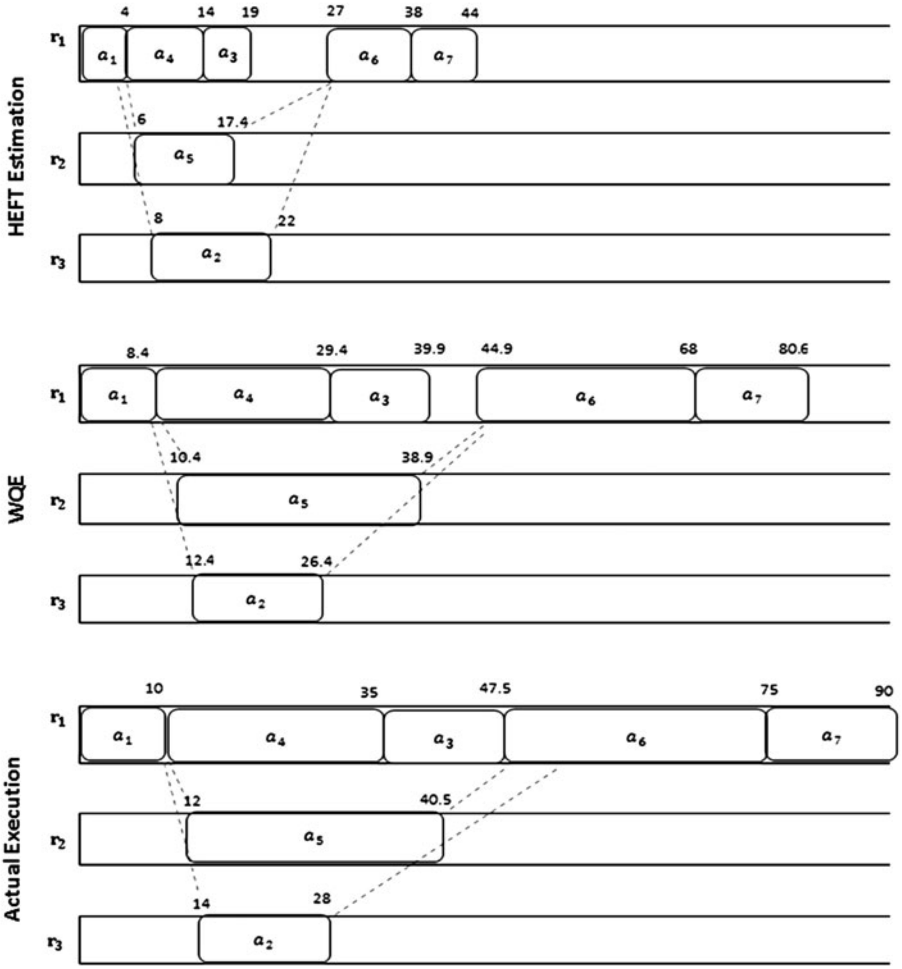


Fig. 14 WQE and HEFT estimation of task execution on resources when the workflow in Fig. 12 has been submitted at time 70. The actual execution has been shown for comparison

"*seq*", "sel" or "*par*" and function as operators. In order to maximize the middle nodes, the least number of operands, which is obviously two, should be selected for each node. Consequently, the biggest tree is a full binary tree having M leaves. This tree can be balanced or unbalanced. As analyzing either case gives the same result, we continue the analysis with a balanced full tree, i.e., full and complete tree. Let M' be the smallest positive number greater than or equal to M such that it is a power of two. In other words, $M' = 2^{\lceil \log_2^M \rceil}$. Assuming these M' nodes as leaves of a tree, the total number of nodes in a full and complete tree is $2^{\log_2^{M'}+1} - 1 = 2M' - 1$. Above each node of this tree, a "*loop*" can be inserted which leads the number of nodes being $2 \times (2M' - 1) = O(M)$. □

Table 3 WQE and HEFT results for workflow of Fig. 12

Submit time: 70

HEFT Estimation

Reliability: 1.0

Makespan: 44.0

cost: 4.51

WQE

$\pi_{1,\text{available}} = 0,\ \pi_{1,\text{user present}} = 0.37,\ \pi_{1,\text{CPU thr exceeded}} = 0.63,\ \pi_{1,\text{unavailable}} = 0$

$w_{1,i\in\{0..50\}} = \frac{10.5}{70-\frac{0+50}{2}} = 0.23$ $\quad w_{1,i\in\{50..69\}} = \frac{10.5}{70-\frac{50+69}{2}} = 1$

$\pi_{1,\text{user present}} = \frac{51\times 0.23}{51\times 0.23+20\times 1} = 0.37$ $\quad \pi_{1,\text{user present}} = \frac{20\times 1}{51\times 0.23+20\times 1} = 0.63$

$\pi_{2,\text{available}} = 0,\ \pi_{2,\text{user present}} = 0,\ \pi_{2,\text{CPU thr exceeded}} = 1,\ \pi_{2,\text{unavailable}} = 0$

$\pi_{3,\text{available}} = 1,\ \pi_{3,\text{user present}} = 0,\ \pi_{3,\text{CPU thr exceeded}} = 0,\ \pi_{3,\text{unavailable}} = 0$

Reliability: 1.0

Makespan: 80.67

cost: 2.41

Actual Execution ⟨Successful execution⟩

Makespan: 90.0

cost: 2.01

Submit time: 250

HEFT Estimation

Reliability: 1.0

Makespan: 44.0

cost: 4.5

WQE

$\pi_{1,\text{available}} = 0,\ \pi_{1,\text{user present}} = 0.07,\ \pi_{1,\text{CPU thr exceeded}} = 0.35,\ \pi_{1,\text{unavailable}} = 0.58$

$\pi_{2,\text{available}} = 0,\ \pi_{2,\text{user present}} = 0,\ \pi_{2,\text{CPU thr exceeded}} = 1,\ \pi_{2,\text{unavailable}} = 0$

$\pi_{3,\text{available}} = 1,\ \pi_{3,\text{user present}} = 0,\ \pi_{3,\text{CPU thr exceeded}} = 0,\ \pi_{3,\text{unavailable}} = 0$

Reliability: 0.013

$r(par\{a_2,\ldots,a_5\}, \boldsymbol{M}) = (0+0.07+0.35)^2 \times 1^2 = 0.176$

$r(root, \boldsymbol{M}) = r(a_1, \boldsymbol{M}) \times r(par, \boldsymbol{M}) \times r(a_6, \boldsymbol{M}) \times r(a_7, \boldsymbol{M}) = 0.42^3 \times 0.176 = 0.013$

Makespan: 85.51

cost: 2.18

$c(a_1, \boldsymbol{M}) = 40 \times 10^{-2} \times \{\frac{0.07}{0.42} \times \frac{7}{10} + \frac{0.35}{0.42} \times \frac{4}{10}\} = 0.18$ $\quad c(a_2, \boldsymbol{M}) = 70 \times 10^{-2} \times \{\frac{1}{1} \times \frac{5}{10}\} = 0.35$

$c(a_3, \boldsymbol{M}) = 50 \times 10^{-2} \times 0.45 = 0.22$ $\quad c(a_4, \boldsymbol{M}) = 100 \times 10^{-2} \times 0.45 = 0.45$

$c(a_5, \boldsymbol{M}) = 80 \times 10^{-2} \times \{\frac{1}{1} \times \frac{2.8}{10}\} = 0.22$ $\quad c(a_6, \boldsymbol{M}) = 110 \times 10^{-2} \times 0.45 = 0.49$

$c(a_7, \boldsymbol{M}) = 60 \times 10^{-2} \times 0.45 = 0.27$ $\quad c(root, M) = 2.18$

Actual Execution ⟨Failure execution⟩

As our analysis has shown the same time complexity for the response time, reliability and cost estimation, in the rest of this section we concentrate on the response time estimation. Estimating the response time involves four processing steps: first, the availability prediction of resources, second, data transfer time computation, third, the response time estimation of activities, and fourth, the response time composition.

Let $|SI|$ be the number of subintervals within prediction window of size L. Note[3] that $1 \leq |SI| \leq L$. The availability prediction is done by processing the subintervals within a window.[4] Thus, the availability prediction for all resources is $O(|SI| \cdot N)$.

To compute the mean data transmission time for each pair of activities, a computation as in Eq. (5) is required for each edge in the workflow. According to Theorem 5.1, this processing needs $O(M)$ operations.

The estimation of response time of activity a_i needs $O(|\operatorname{pred}(a_i)|)$ operations to compute the latency of data transmission time and $O(|AL|)$ time to compute the weighted sum of the response time at availability levels as indicated by Eq. (11). In total, the estimation of the response time for activity a_i takes[5] $O(|\operatorname{pred}(a_i)|)$ operations.

The composition is done via a post order traversal of the corresponding tree in which the complexity depends on the number of nodes in the tree. According to Theorem 5.2, the process is done within $O(M)$ time.

The time complexity of all four steps results in $O(|SI| \cdot N + M + \sum_{i=1,\ldots,M} |\operatorname{pred}(a_i)| + M) = O(|SI| \cdot N + M + E) \overset{\text{Theorem 5.1}}{\Longrightarrow} O(|SI| \cdot N + M)$ where $1 \leq |SI| \leq L$.

6 Experiments

The simulation has been done over NDU data set. Condor has recorded the availability changes of 64 nodes over 6 months in the early 2007 [21]. To choose a suitable resource state predictor, in the first part of the simulation, PE and PW with different window sizes are initially examined. The results in the second part of the simulation are based on the best performed predictor. To measure the accuracy of the workflow QoS estimation, the actual QoS parameters have been gained by simulating the execution of workflows on resources which behave exactly according to the NDU trace. The errors of the QoS estimation in HEFT and the WQE methods have been measured in comparison with the actual values.

6.1 Resource state prediction results

The state of each resource has been predicted each hour within 6 months. The availability state with the maximum probability has been selected as output of the predictor as in Eq. (15):

$$\text{predicted state for resource } j \text{ at time } t = \arg\max_{al} \pi_{j,al}(t) \tag{15}$$

[3] In the case that $|SI| = 1$, all observations refer to one availability state. In the case that $|SI| = L$, each observation refers to a different availability state in comparison with its previous and next observation.

[4] There is no difference between complexity of PW and PE. PW scans subintervals twice while PE scans once.

[5] As the number of availability levels is bounded, it can be ignored in complexity computation.

Fig. 15 Resource availability prediction accuracy vs. size of window

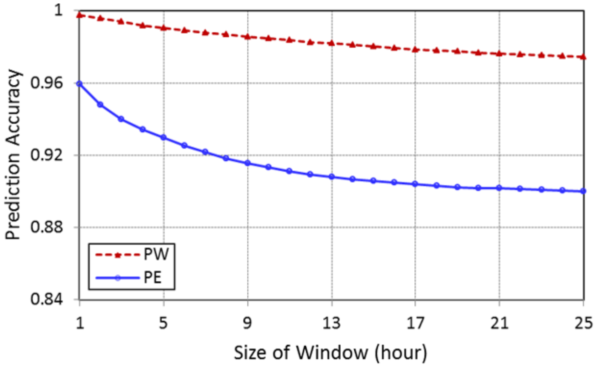
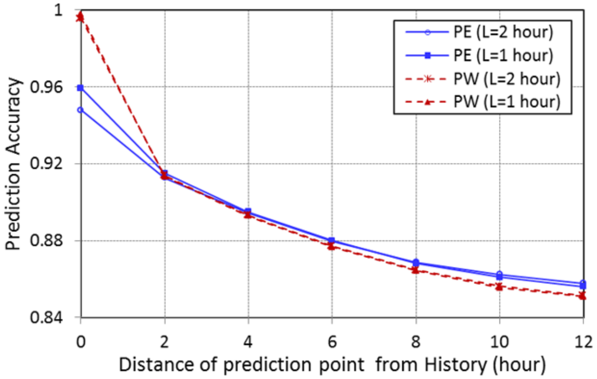


Fig. 16 Resource availability prediction accuracy vs. distance of prediction point from history



When the predicted state is equal to the actual reported state in the trace, the prediction is correct otherwise it is wrong. Figure 15 indicates the prediction accuracy when the window size changes from 1 to 25 hour. The smaller the window size, the better the achieved accuracy. PW outperforms PE. The reason is that the information near to the prediction time gets more weight in PW. This makes sense as the state of a resource is more dependent on its recent states rather than farther states in the past.

To examine the effect of distance of prediction point from history, we have chosen 1 and 2 hour as the window size, since, in Fig. 15, they showed the highest accuracy. Figure 16 shows how the accuracy reduces when the distance of prediction point from history increases. This accuracy reduction has a great effect in workflow QoS estimation. After submitting the workflow, it takes some time for the activities to become ready for execution. So, when we predict the state of a resource at a workflow submit time, the prediction does not necessary hold true when the activity starts execution. The figure shows that when the distance of prediction point changes from 0 to 12 hours, the accuracy reduces from 96 % to 85 %. When this distance is less than 2 hours, the PW performs better, but for distances above 2 hours PE has performed better. In the rest of the experiments, we use PW with window size of 1 hour as a predictor, since 70 % of the workflows have makespan less than 4 hours and in this range PW performs better than PE.

Table 4 Parameters related to resources and 25620 random generated workflows in the simulation

Category	Parameter Name	Value
Workflows	Number of activities	$\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70\}$
	Height	$\{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$
	And–Or ratio	$\{0, 0.2, 0.4, 0.6, 0.8, 1\}$
	CCR	$\{0, 0.2, 0.4, 0.6, 0.8, 1, 1.2\}$
	Max edge ratio	$\{1.1, 1.3, 1.5, 1.7, 1.9\}$
	Operations of activity	$N\{\mu = U(15\text{ MOP}, \ldots, 1500\text{ MOP}), \sigma = 0.25\mu\}$
Resources	*RE*	56 nodes of NDU trace[a]
	CS_j	Floating point operation speed of resource j in NDU trace
	α_{al}	$i_{al} \times 0.3$
	β_{cost}	10^{-7}

[a] Nodes with ids 14, 32, 56, 4, 15, 61, 42, 26 have been omitted to increase load balancing among resources

6.2 Workflow QoS estimation results

6.2.1 Random workflows

To measure the accuracy of QoS estimation, 25620 random workflows were generated. The 56 nodes of NDU trace have been considered as resources. Simulation parameters have been shown in Table 4. Among the parameters, *Height* shows the height of the workflow graph which is equal to the length of the longest path from the root to the exit activity. *And–Or ratio* is the probability that the basic structure would be *and* rather than *or* when a split occurs. *Max edge ratio* is a parameter controlling the maximum out-degree of a node. The number of operations per activity is modeled by a normal distribution with its mean uniformly selected from the range of 15 Mega Operation (MOP) to 1500 MOP while the standard deviation is a quarter of the mean. For the sake of simplicity, we have assumed that there is no loop in the workflows. As there was no bandwidth information for resources of NDU trace, we use the *C*ommunication-to-*C*omputation *R*atio (CCR) to compute communication time [19]. The communication time between each two activities is modeled as a normal distribution with mean computed by Eq. (16) and the standard deviation of 20 s. We assume that network communication is reliable.

$$\mu\, dt(a_i, a_j, \boldsymbol{M}) = \begin{cases} 0 & \text{if } M(a_i) = M(a_j) \\ \text{CCR} \times \dfrac{\frac{o_i}{CS_{M(ai)}} + \frac{o_j}{CS_{M(aj)}}}{2} & \text{else} \end{cases} \tag{16}$$

In this part of simulation, we compare the accuracy of HEFT estimation of QoS with WQE estimated values. The accuracy is computed by comparing the estimated values with actual values achieved by simulating workflow execution on NDU trace. Each workflow is submitted to the grid in time interval of 2 hours during 6 months. The results are the average for all workflows and all submit times.

Table 5 shows the failure and the resource utility reports of workflows execution. It has been found that about 75 % of workflows have a failure rate less than 60 %,

Table 5 Failure and resource utility report

Failure percent		[0, 20 %]	(20 %, 40 %]	(40 %, 60 %]	(60 %, 80 %]	(80 %, 100 %]
Failure report	Percent of workflows	21 %	29 %	25 %	13 %	12 %
	Total mean	43 %				
Resource utilization		[0, 20 %]	(20 %, 40 %]	(40 %, 60 %]	(60 %, 80 %]	(80 %, 100 %]
Resource utility report	Percent of workflows	54 %	25 %	11 %	4 %	6 %
	Total mean	27 %				

Table 6 Mean actual response time and cost report

Actual response time (hour)		(0, 2]	(2, 4]	(4, 6]	(6, 8]
Response time report	Percent of workflows	30 %	40 %	27 %	3 %
	Total mean	3			
Actual cost (base unit)		(0, 1425]	(1425, 2850]	(2850, 4275]	(4275, 5700]
Cost report	Percent of workflows	44 %	35 %	16 %	5 %
	Total mean	1827			

while 25 % of them have a failure rate above 60 %. About 80 % of workflows use less than 40 % of the resources. The reason is that HEFT is a greedy algorithm toward using the fastest resources which is defined by the static speed of resources.

Table 6 illustrates the actual response time and cost report of successfully executed workflows, respectively. 70 % of workflows have been executed within 4 hours, and the execution of the rest has been completed in at most 8 hours. About 80 % of the workflows have been executed with the cost less than 2850 of base units and the rest of them have cost up to 5700.

In order to measure the accuracy of reliability prediction, we use the following rules:

$$\begin{aligned}
&\textbf{rule 1}: && \text{if workflow execution failed} \wedge r\big(root(T), \boldsymbol{M}\big) < 0.5 \ \rightarrow \text{accurate}\\
&\textbf{rule 2}: && \text{if workflow execution succeeded} \wedge r\big(root(T), \boldsymbol{M}\big) > 0.5 \ \rightarrow \text{accurate}\\
&\textbf{rule 3}: && \text{otherwise} \rightarrow \text{inaccurate}
\end{aligned} \tag{17}$$

When the reliability is predicted correctly, we measure the certainty of prediction by confidence value as in Eq. (18):

$$confidence = \begin{cases} 1 - r(root(T), \boldsymbol{M}) & \textbf{rule 1} \\ r(root(T), \boldsymbol{M}) & \textbf{rule 2} \end{cases} \tag{18}$$

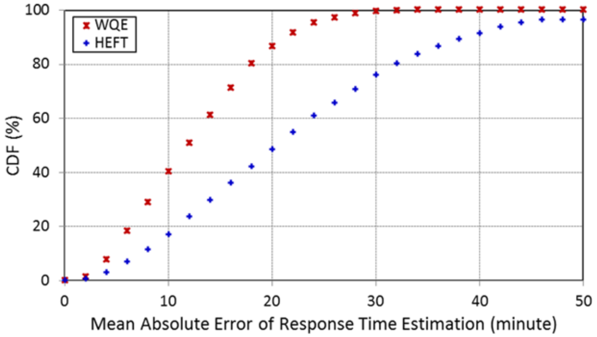


Fig. 17 Cumulative distribution function of mean absolute error for response time estimation

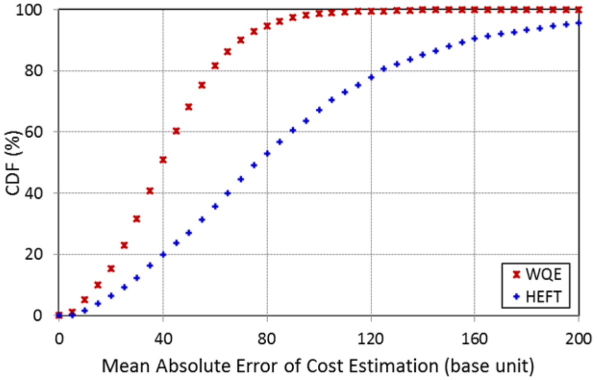


Fig. 18 Cumulative distribution function of mean absolute error for cost estimation

85 % of failure/success execution of workflows has correctly been predicted in WQE with confidence of 0.82. HEFT has the accuracy of 56 % in failure/success prediction which is low in comparison with WQE. This is because HEFT always assumes that the execution will be completed successfully.

The cumulative distribution function (CDF) of the mean absolute error (MAE) of the response time estimation has been shown in Fig. 17. The curve of WQE is above HEFT, which indicates its superior performance due to considering resource state prediction in the computation of the QoS parameters. In 80 % of cases, the MAE in WQE is less than 18 minutes, while this value increases to 32 minutes in HEFT. Similar results have been obtained in cost prediction as shown in Fig. 18.

In the rest of this section, we investigate the effect of the workflow structure on estimation accuracy. As the behavior of cost curves was similar to the response time curves, due to lack of space, we only show the results of the response time estimation.

Figure 19 indicates the effect of height when it varies in the range from 2 to 20. As it is shown, the increase of height causes a makespan increment, and therefore, an increment in distance of prediction point from history. Thus, the quality of estimation decreases. In this way, a slight descending slope in the accuracy of reliability prediction and at the same time a large ascending slope in MAE of the response time are generated. HEFT has reverse behavior in reliability prediction. Since, when

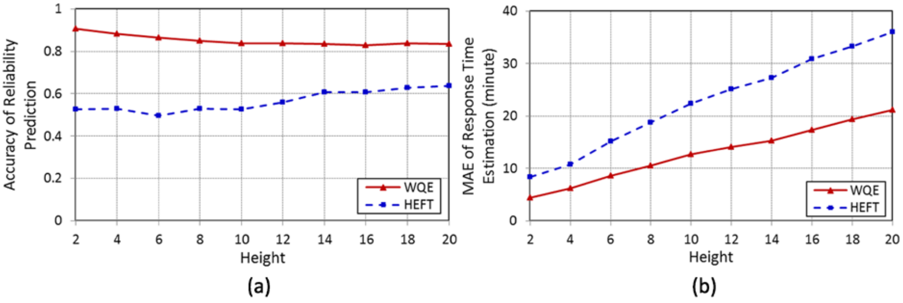


Fig. 19 The effect of the height variation on the accuracy of workflow QoS estimation. The accuracy of reliability is the percentage of times that a success/failure execution of a workflow has been correctly predicted using the rules in Eq. (17)

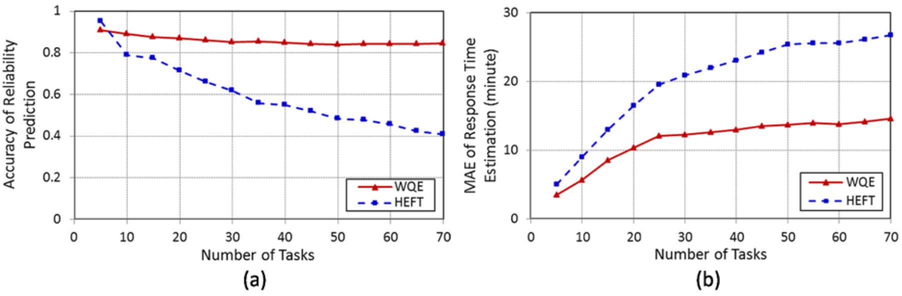


Fig. 20 The effect of the number of tasks variation on the accuracy of a workflow QoS estimation. The accuracy of reliability is the percentage of times that a success/failure execution of a workflow has been correctly predicted using the rules in Eq. (17)

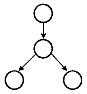
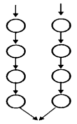
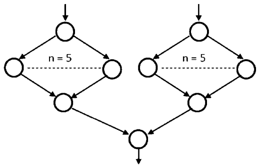
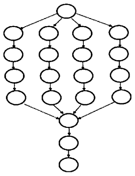
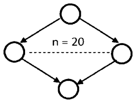
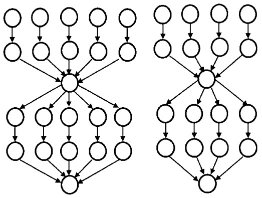
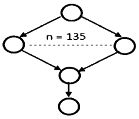
the height increases,[6] it becomes greedier toward using few fast resources. In NDU trace, fast resources have high availability. Thus, less reliability prediction error will be involved in HEFT.

To investigate the effect of the number of tasks, we have shown the accuracy as a function of the number of tasks in the range of $\{5, 10, \ldots, 70\}$. Figure 20 shows the result. As expected, HEFT performs worse with a considerable slope variation. On the other hand, we get a less variable slope behavior in WQE. The reason is that increasing the number of tasks does not necessary make an increment in the makespan. It means that the distance of prediction point is kept rather at the same level. So, in WQE, the quality of estimation does not change a lot.

Figures 19 and 20 also show that as the workflow becomes larger either by height or the number of tasks, WQE will outperform HEFT even more. In large workflows, HEFT causes a huge error in estimation while WQE has much better performance due to regarding the dynamic states of resources in QoS estimation. For very small

[6] Since we are assuming approximately the same amount of tasks, the workflow gets closer to the shape of a sequence structure.

Table 7 Actual workflow structures used in simulation

	
(a) Avian Flu	(b) Gene2Life
	
(c) PSLoad	(d) Epigenomics
	
(e) Animation Workflow	(f) Ligo Inspiral Analysis
	
(g) Motif	

workflows, for example, with height less than 4 as shown in Fig. 19(b) or under 10 activities as shown in Fig. 20, applying WQE has less benefit.

6.2.2 Actual workflows

The method has been evaluated on actual workflows (Table 7). We have constructed the tree according to the structure of workflows. The CCR is supposed to be 0.2 to enhance the effect of resource dynamicity on QoS parameters. The number of operations per activity is a normal distribution with mean uniformly chosen from 15 MOP to 1500 MOP and the variance equal to a quarter of the mean. Table 8 shows the results. The superior performance of WQE is enhanced when the workflow becomes bigger. For example, for Avian Flu, the MAE of the response time estimation in WQE is about 79 s less than HEFT, while in Motif, this outperformance increases to 39 min. For very small workflows like Avian Flu and Gene 2 Life, the accuracy of reliability prediction in WQE is less than HEFT, but for other workflows the WQE has predicted better. The reason is that small workflows consume few fast resources.

Table 8 Evaluation of QoS estimation on actual workflows

Workflow	Number of tasks	Resource utility	Failure percent	Mean actual response time (hour)	Mean actual cost	WQE				HEFT		
						Reliability accuracy	Reliability confidence	MAE of response time (min)	MAE of cost	Reliability accuracy	MAE of response time (min)	MAE of cost
Avian Flu [32]	4	3.6 %	0.07	0.95	285	0.89	0.88	2.38	5.3	0.93	3.70	10.6
Gene 2 Life [32]	8	3.6 %	0.07	1.43	648	0.87	0.86	3.6	10.8	0.93	5.43	19.7
PSLoad [32]	20	16.0 %	0.50	1.65	1641	0.81	0.79	7.10	25.4	0.50	10.81	47.1
Epigenomics [33]	20	7.1 %	0.35	2.64	1438	0.88	0.88	7.26	21.1	0.64	14.8	48.8
Animation workflow [34]	22	19.6 %	0.55	1.63	1717	0.79	0.78	3.80	20.9	0.45	10.8	58.6
Ligo inspiral analysis [33]	40	21.4 %	0.55	2.63	2879	0.79	0.78	5.91	29.3	0.45	15.9	94.1
Motif [32]	138	100 %	0.95	2.61	8530	0.95	0.95	15.5	115.4	0.04	54.3	481.1

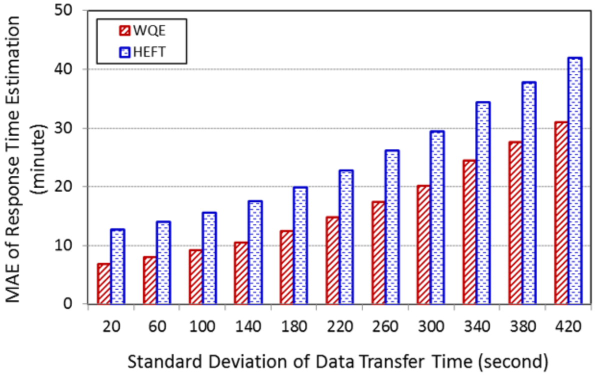


Fig. 21 The effect of the data transfer time variation on response time estimation

These resources in NDU trace are also highly available. When the size of workflow grows, WQE provides more accurate prediction for reliability.

6.2.3 Network stability effect

In the previous sections, we assumed that the data transmission time between each pair of activities is done with a mean computed by Eq. (16) and the standard deviation of 20 s. As the network may not be stable for communications, this variance might be higher in reality. To investigate the effect of this variation, we have changed the standard deviation of data transmission time for each edge in the Epigenomics workflow in the range from 20 to 420 s. The result is shown in Fig. 21. As the standard deviation increases, the estimation error of both methods increases, since the estimation of communication time has a direct effect on workflow QoS estimation. A good prediction method for network operation can be composed with the proposed method to minimize the effect of data transfer time variation.

6.2.4 Computation time

In this part of simulation, the effect of the parameters on the run time of WQE has been investigated. These parameters include the number of activities (M), number of resources (N), and prediction window size (L). In each part of the simulation, one parameter changes while others remain constant. A Java-based simulator runs on a system with Intel® Core™ i7-3770k CPU (3.50 GHz) and uses 128 MB of memory.

In the first part of the simulation, $N = 64$ and $L = 1$ h. M changes in the range of $\{500, 1000, \dots, 8500\}$ to reflect various numbers of activities. To have fair communication-to-computation and the number of parallel-to-selection splits, CCR and And–Or ratio have both been selected to be 0.5. The max edge ratio is 1.5 and the height is 50. Figure 22 shows that the run time of the algorithm changes from 17.2 ms for a workflow with 500 activities to 68.6 ms for a workflow with 8500 activities. As expected by theoretical analysis of Sect. 5, there is a linear relationship among the points of the plot. For example, when the number of activities changes from 1000 to 4000, the run time smoothly increases from 28.8 to 45.7 ms.

Fig. 22 Run time of WQE for workflows with different sizes. CCR = 0.5; And–Or ratio = 0.5; Max edge ratio = 1.5; height = 50; $L = 1$ h. NDU resources have been used. The results are averages over 50 runs

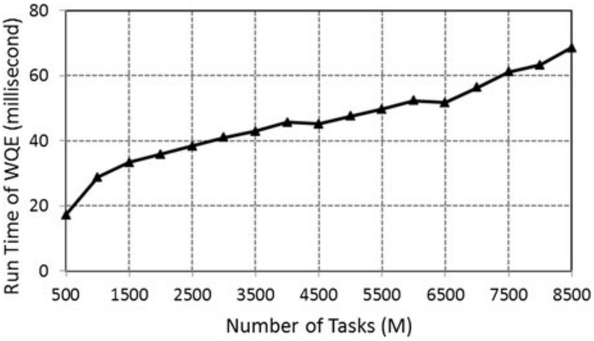


Fig. 23 Run time of WQE for workflow with 4500 tasks vs. number of resources. CCR = 0.5; And–Or ratio = 0.5; Max edge ratio = 1.5; height = 50; $L = 1$ h. The results are averages over 50 runs

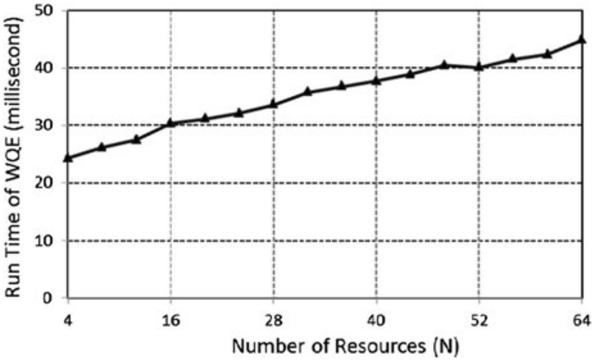
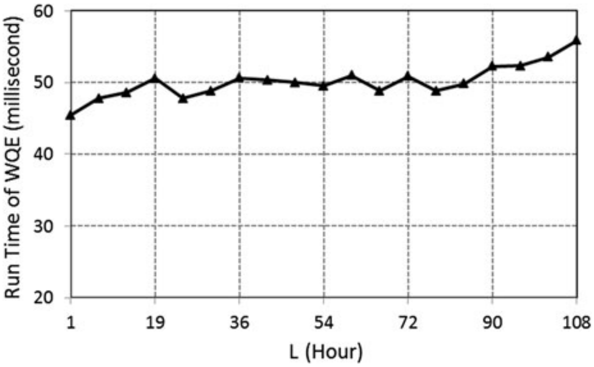


Fig. 24 Run time of WQE for workflow with 4500 tasks vs. size of window. CCR = 0.5; And–Or ratio = 0.5; Max edge ratio = 1.5; height = 50. The results are averages over 50 runs



In the second part of the simulation, $M = 4500$ and $L = 1$ h. N changes in the range of $\{4, 8, \ldots, 64\}$ to reflect various numbers of resources. As expected by theoretical analysis, there is a linear relationship among the run time and the number of resources as shown in Fig. 23.

Finally, we change L in the range of $\{1, 7, \ldots, 108\}$ hours in the case when $M = 4500$ and $N = 64$. Figure 24 shows that as the size of the window increases, the run time of the algorithm slightly changes from 45.4 to 55.9 ms. The slight changes indicate that the most important parameters in the run time of WQE are the number of activities in the workflow and the number of resources. Increasing the size of the prediction window slightly increases the number of subintervals and thus has less effect on the run time increment.

7 Conclusion and directions for future work

Accurate workflow QoS estimation enhances the performance of a workflow scheduling algorithm. In this paper, we propose a method called WQE, for estimating the QoS parameters of a Grid Workflow. These parameters include reliability, response time and execution cost. The two main components of WQE include resource monitoring and analysis and workflow QoS computation.

We have employed a simple monitoring system which monitors each resource periodically to record the changes in availability state of them within time. The availability states represent both the workload and availability of resources. The resource behavior with respect to availability changes is modeled by a multi-state system. Two prediction algorithms (PE and PW) have been proposed to stochastically predict the availability state of a resource. These predictors use different weighting mechanisms for historical availability information. Simulation results showed the superior performance of PW in comparison with PE.

The workflow QoS computation is done in four steps: data transfer modeling, activity level estimation, basic structures computation, and QoS composition. The QoS of activities are computed based on resources availability analysis. We support sequential, parallel, selection, and loop as basic structures. The QoS of each basic structure is computed by aggregating the QoS of each operand involved in the basic structure. Assuming the workflow graph is converted to a tree structure, the QoS composition uses QoS of basic structures to compute the QoS of the root which is regarded as the final computation. NDU trace has been used to simulate workflow executions to get the actual QoS values. Simulations have been carried on for random and actual workflows. WQE outperforms estimation of HEFT, and the estimated values are much closer to actual values.

There are three directions for future work. First, the presented estimation method can be exploited to enhance the quality of a workflow scheduling algorithm. A good trade-off among reliability, performance, and cost in scheduling is possible when employing estimated QoS of workflow. Second, a network operation prediction method can be combined with WQE to improve consistency with the actual world. Finally, the method might be justified for estimating the QoS of the workflow running on virtual machines inside a data center to move toward cloud computing. The major challenge will be predicting the behavior of a virtual machine which needs much more investigation in this way.

Acknowledgements This work is being supported by Iran Telecommunication Research Center (ITRC), Tehran, Iran, Contract No. 12200/500.

References

1. EGEE homepage (2008). http://egee.cesnet.cz/en/info/
2. Teragrid homepage. http://www.teragrid.org

3. PlanetLab (2008) P.L.A. open platform for developing debugging and accessing planetary scale services. http://www.planet-lab.org
4. Frey J, Tannenbaum T, Livny M, Foster I, Tuecke S (2001) Condor-g: a computation management agent for multi-institutional grids. In: International conference on high performance distributed computing, pp 55–63
5. Litzkow M, Livny M, Mutka M (1988) Condor—a hunter of idle workstations. In: International conference on distributed computing systems, pp 104–111
6. Rood B, Lewis MJ (2009) Grid resource availability prediction-based scheduling and task replication. J Grid Comput 7:479–500
7. Kiran M, Hashim A-HA, Kuan LM, Jiun YY (2009) Execution time prediction of imperative paradigm tasks for grid scheduling optimization. Int J Comput Sci Netw Secur 9:155–163
8. Smith W (2007) Prediction services for distributed computing. In: International symposium on parallel and distributed processing, pp 1–10
9. Tao M, Dong S, Zhang L (2010) A multi-strategy collaborative prediction model for the runtime of online tasks in computing cluster/grid. Clust Comput 14:199–210
10. Glasner C, Volkert J (2011) Adaps—a three-phase adaptive prediction system for the runtime of jobs based on user behaviour. J Comput Syst Sci 77:244–261
11. Byun E, Choi S, Baik M, Gil J, Park C, Hwang C (2007) MJSA: Markov job scheduler based on availability in desktop grid computing environment. Future Gener Comput Syst 23:616–622
12. Ramakrishnan L, Reed D (2009) Predictable quality of service atop degradable distributed systems. Clust Comput. doi:10.1007/s10586-009-0078-y
13. Wang H-C, Lee C-S, Ho T-H (2007) Combining subjective and objective QoS factors for personalized web service selection. Expert Syst Appl 32:571–584
14. Hwang S-Y, Wang H, Tang J, Srivastava J (2007) A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. Inf Sci 177:5484–5503
15. Jaeger MC, Rojec-Goldmann G, Muehl G (2004) QoS aggregation for web service composition using workflow patterns. In: International conference on enterprise distributed object computing, pp 149–159
16. Zheng H, Yang J, Zhao W (2010) QoS probability distribution estimation for web services and service compositions. In: International conference on service-oriented computing and applications, pp 1–8
17. Maheswaran M, Ali S, Siegel HJ, Hensgen D, Freund RF (1999) Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. J Parallel Distrib Comput 59:107–131
18. Topcuoglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13:260–274
19. Cao H, Jin H, Wu X, Wu S, Shi X (2010) DAGMap: efficient and dependable scheduling of DAG workflow job in grid. J Supercomput 51:201–223
20. Chen WN, Zhang J (2009) An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. IEEE Trans Syst Man Cybern 39:29–43
21. http://fta.scem.uws.edu.au/
22. Wolski R, Spring N, Hayes J (1999) The network weather service: a distributed resource performance forecasting service for metacomputing. Future Gener Comput Syst 15:757–768
23. Dinda P, O'Hallaron D (1999) An extensive toolkit for resource prediction in distributed systems. Technical report CMU-CS-99-138, Carnegie Mellon University
24. Hu L, Che X-L, Zheng S-Q (2012) Online system for grid resource monitoring and machine learning-based prediction. IEEE Trans Parallel Distrib Syst 23:134–145
25. Jiong Y, Guo-Zhong T, Ling C (2008) Allocating resource in grid workflow based on state prediction. In: IEEE/IFIP international conference on embedded and ubiquitous computing, pp 417–422
26. Lili S, Shoubao Y (2009) A Markov chain based resource prediction in computational grid. In: International conference on frontier of computer science and technology, pp 119–124
27. Ren X, Lee S, Eigenmann R, Bagchi S (2007) Prediction of resource availability in fine-grained cycle sharing systems empirical evaluation. J Grid Comput 5:173–195
28. Wu AS, Yu H, Jin S, Lin K-C, Schiavone G (2004) An incremental genetic algorithm approach to multiprocessor scheduling. IEEE Trans Parallel Distrib Syst 15:824–834
29. Tao F, Zhao D, Hu Y, Zhou Z (2008) Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system. IEEE Trans Ind Inform 4:315–327
30. Young L, McGough S, Newhouse S, Darlington J (2003) Scheduling architecture and algorithms within the ICENI grid middleware. In: UK e-science all hands meeting, pp 5–12

31. Cardellini V, Casalicchio E, Grassi V, Iannucci S, Presti FL, Mirandola RR (2012) MOSES: a framework for QoS driven runtime adaptation of service-oriented systems. IEEE Trans Softw Eng 38:1138–1159
32. Ramakrishnan L, Plale B (2010) A multi-dimensional classification model for scientific workflow characteristics. In: International workshop on workflow approaches to new data-centric science
33. Pegasus Workflow Generator. http://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator
34. Chong A, Sourin A, Levinski K (2006) Grid-based computer animation rendering. In: International conference on computer graphics and interactive techniques, pp 39–47