A Probabilistic Batch Oriented Proactive Workflow Management

Panagiotis Oikonomou
Comp. Science and Telecommunications
University of Thessaly
Lamia, Greece
paikonom@uth.gr

Kostas Kolomvatsos
Comp. Science and Telecommunications
University of Thessaly
Lamia, Greece
kostasks@uth.gr

Christos Anagnostopoulos
School of Computing Science
University of Glasgow
Glasgow, UK
christos.anagnostopoulos@glasgow.ac.uk

Nikos Tziritas
Comp. Science and Telecommunications
University of Thessaly
Lamia, Greece
nitzirit@uth.gr

Georgios Theodoropoulos
Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
georgios@sustech.edu.cn

***Abstract*—Workflow management is a widely studied research subject due to its criticality for the efficient execution of various processing activities towards concluding innovative applications. A set of models has been already proposed dealing with finding the most appropriate node to conclude the placement of each task present in a workflow. The ultimate goal is to eliminate the required time for delivering the final outcome taking into consideration the dependencies between tasks. In this paper, we go a step forward and enhance the decision making of a scheduler with a batch oriented approach to deal with a high number of workflows. We also focus on a gap of the respective literature, i.e., apart from the time and cost requirements, we focus on the statistics of the underlying data where tasks should be executed. We provide a probabilistic 'data' oriented approach combined with a 'infrastructure' oriented scheme to pay attention on dynamic environments where the underlying data are continuously updated trying to minimize the network overhead for migrating data. We propose the sequential management of workflows, i.e., we map the workflows requirements for data with the available datasets, then, combine the outcome with an optimization model upon the time requirements and the cost of every placement. The performance of our sequential management is revealed by a high number of experiments depicting the advantages in the network overhead. Our evaluation deals with a high number of real workflow applications and a comparative assessment with other baseline schemes.**

***Index Terms*—workflows management, tasks scheduling, decision making, probabilistic model, distributed processing**

I. Introduction

The mapping of a number of connected tasks upon a set of heterogeneous resources/nodes is the subject of workflow scheduling. The problem is widely studied in the research community being active in the domains of Grid and Cluster Computing [1]. Recently, one can observe an increased interest in workflow scheduling in Cloud [2]. There, we can detect a substantial amount of resources, a variety of virtual platforms and enjoy zero cost for management/maintenance. Various challenges should be met before we are in a position to adopt fully automated services for workflow management. For instance, the pay-as-you go model and data-transfer costs can be a obstacle to Cloud's potentials [1].

Utilizing processing nodes at the Cloud involves a monetary cost that depends on the time devoted to reserve the resources. Additionally, the performance of resources varies due to the resource sharing of Virtual Machines (VMs). For instance, Amazon Web Services (AWS) offers the following categories: (i) on-demand with fixed price per time unit; (ii) reserved resources with a lower price than the on-demand service and a long lease; (iii) spot instances dealing with the reservation of unused capacity and a significant discount. Similar strategies are adopted by other providers like Google, Alibaba, Microsoft, etc. We can easily detect the trade off between acquiring a reserved resource and pay more compared to the acquisition of spot instances with a significantly reduced cost but with no guarantee for the success of the reservation. Obviously, the target of users is to execute their workflow applications in the minimum time and cost. The minimization of the execution time, i.e., makespan, is a research subject that has been extensively studied by the research community and a set of algorithms are proposed like HEFT, CPOP [3] and DCP [4]. However, such algorithms neglect the increased cost incurred by placing tasks without taking into consideration the cost incurred by the transfer of data by external nodes or resources. This cost is realized by the network overhead imposed by selecting processing nodes owing datasets with limited similarity with the data demanded by workflows. To the best of our knowledge, the majority of the relevant efforts in the domain, they target to 'infrastructure' oriented decisions paying limited attention on the data upon which every workflow should be executed. Evidently, workflow tasks may demand for specific data to perform the envisioned processing. The assumption behind other models is that the requested data can be present at any processing node or they can be migrated to the node where the execution will be realized. However, data migration incurs an increased network overhead.

In this paper, we argue upon a holistic model that takes into consideration not only the 'infrastructure oriented' parameters but also the 'data oriented' aspects. We investigate a dynamic scheduling approach and resource provisioning management like in our past efforts [5], [6], however, we focus on the data required by tasks and match them against the data present at the processing nodes before we conclude any placement. We propose the use of two orchestrators, i.e., the Global Orchestrator (GO) and the Local Orchestrator (LO). The GO is responsible to reveal the distance between the data requested by a workflow and the available datasets. The GO relies on a probabilistic model and performs the 'data' oriented scheduling and it realizes the first step of our workflow sequential management. The GO delivers an initial placement, then, as multiple workflows may be assigned in the same node, the LO proceeds to refinements and concludes the prioritization of the tasks. The LO separates the incoming tasks in two categories based on their criticality and proceeds by placing them for execution using the Shortest Job First (SJF) policy subject to the decisions made by the GO. The intuition behind our approach is two-fold: At first, we seek to maximize the data mappings and, secondly, to minimize the infrastructure oriented parameters and limit the execution time and cost. To the best of our knowledge, this is one of the first attempts that deal with the combination of infrastructure and data oriented parameters in the scheduling of scientific workflows. The following list reports on the contribution of our work:

- We propose two orchestrators, i.e., the GO and the LO to perform the scheduling at two levels;
- For supporting the GO, we provide a probabilistic model for exposing the distance between workflows requirements and data present at every processing node;
- For supporting the LO, we provide a fast approach that improves tasks throughput making sure that shorter jobs are executed first chasing a short turnaround time;
- We perform an extensive experimental evaluation of the proposed model and simulate the execution of a high number of real-world scientific workflows;

The rest of the paper is organized as follows. Section II discuss the prior work in the specific domain. System model and problem formulation are illustrated in Section III. The proposed approach is presented in Section IV. The evaluation of the model is discussed in Section V. Our conclusions and plans for future research are presented in Section VI.

II. Related Work

A decentralized approach for workflow scheduling is presented in [7] where the Dynamic Hierarchical Scheduling (DHS) algorithm is proposed. DHS is a two-stage algorithm where in the first stage it operates using the Dynamic Level Scheduling (DLS) [8] algorithm and in the second stage determines the resource within the group that results in the earliest finish time. A planner-guided dynamic scheduling strategy for multiple workflows is featured in [9]. The proposed system consists of three components namely DAG Planner, Job Pool and Executor. The DAG Planner prioritize each task of the workflow locally using the HEFT algorithm [3]. The Executor globally re-prioritizes the jobs in the Job Pool. The task with the highest priority is scheduled to the resource that result in the earliest finish time. The effect of different scheduling policies in heterogeneous distributed real-time systems has been extensively studied in [10]. The proposed policies work in the context of every list scheduling algorithms i.e., a task prioritization phase and a resource selection phase. For the first phase different algorithm can be applied e.g., Earliest Deadline First (EDF). In [11], the Highest Level First (HLF) [12] or the Least Space–Time First (LSTF) [13]. In the second phase the exploitation of schedule holes is initiated using classic bin packing algorithm are applied like first Fit (FF), Best Fit (BF) and Worst Fit (WF). An extension of [10] is presented in [14] where multiple workflows arrive in a multi-tenant environment. Tsai et al. [15] advances the idea of a clustering technique [16] to reduce the communication cost among tasks. Commonly tasks within the same resource bear no communication overhead. In the same fashion as in [10], the exploitation of schedule holes was carried out using the BF algorithm.

Another algorithm elastically adjusts the number of resources for time constrained workflows [17] or an auto-scaling mechanism where budget and deadline are either distributed in each task [18] (sub-deadline, sub-budget) or are assigned to the entire workflow [19]. ToF [20], is a general transformation-based optimization framework compromised by a transformation and a planner component. The first component consists of six transformation operation e.g., merge and split while the second one performs the transformation on the workflow according to the cost optimization function. Dyna [21] is another Workflow as a Service (WaaS) framework. It considers both spot and on-demand instances and offers a probabilistic deadline guarantee. For each task, a configuration plan is generated while, at the runtime, auto-scaling of resources is reinforced. In [22], the authors integrate an Edge Orchestrator (EO) in an Edge-Cloud environment to assign tasks on edge nodes or offloading them to Cloud, as well as to aggregate data produced at the edge. In [23], multiple factors are considered such deadline, location, budget constraint, energy and computation capacity before assigning a task at the edge of the network or the Cloud. That scheduling problem is tackled by the development of a greedy heuristic algorithm. The authors of [24] propose a dynamic multi-level scheduling scheme that decouples data placement from task scheduling. The first level is initiated prior to tasks execution and, in the second level, tasks are assigned to resources depending on the particular strategy and their dependencies. In the third level, a monitoring mechanism is triggered whenever there is a need to move or replicate data. The authors of [25] hold the view that a high latency is caused by transferring large amount of data across multiple nodes. They proposed GASPO, a data placement method that combines Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO).

III. Preliminaries & Problem Formulation

We consider a set of workflows $\mathcal{W} = \{w_1, w_2, \ldots, w_{|\mathcal{W}|}\}$ that has to be scheduled in a set of processing nodes (present at the Cloud or in a Fog/Edge computing environment) $\mathcal{N} = \{n_1, n_2, \ldots, n_{|\mathcal{N}|}\}$. Workflows arrive in the system at different time units and may request for a specific range of data to perform their actions. w_i (the i^{th} workflow) is modeled as a Directed Acyclic Graph (DAG) (T_i,E_i) where T_i denotes its set of heterogeneous tasks and E_i the set of edges. t_{ij} denotes the j^{th} task of w_i assuming a total ordering ($1 \leq j \leq |T_i|$). Also, e_{ijz} denotes a weighted directed edge connecting t_{ij} and t_{iz} (predecessor) and dw_{ijz} is the corresponding weight. A task may be bounded to data that are located in one or more nodes. This makes the problem more complex while it differentiates it from the majority of works that tackle the single or the multiple workflow scheduling problem. Past efforts assume data dependencies only between tasks. Let a_{ijz} denote a precedence constraint between t_{ij} and data DS_{ij} denoting the j^{th} dataset at n_i. t_{ij} must receive DS_{ij} to start its execution. Communication among nodes n_x and n_y occurs at a certain transfer rate (TR_{xy}). Let B_{xy} be a boolean variable with $B_{xy} = 0$ iff $x = y$ otherwise $B_{xy} = 1$. Also, A_{iz} is a binary variable depicting whether t_{iz} requests data from a predecessor t_{ij} ($A_{iz} = 1$) or requests the k^{th} data from n_j ($A_{iz} = 0$). Assuming that t_{iz} is assigned in n_y and t_{ij} in n_x, the temporal cost for transferring data towards t_{iz} is expressed by $R_{iz} = A_{iz} \times \frac{B_{xy} \times dw_{ijz}}{TR_{xy}} + (1 - A_{iz}) \times \frac{B_{yj} \times D_{jk}}{TR_{yj}}$. In case where t_{iz} and t_{ij} are assigned in the same node, then, $R_{iz} = 0$. The same holds true when t_{iz} and D_{jk} are assigned in the same node. Assuming that t_{ij} is assigned for execution to n_z, the Estimated Start Time (EST) of t_{ij} is expressed by $EST^{n_z}_{t_{ij}} = max\{available_z, max_{t_{ik} \in pred(t_{ij})}\{AFT_{t_{ik}} + R_{ij}\}\}$ while the Estimated Finish Time (EFT) by $EFT^{n_z}_{t_{ij}} = ET_{ijz} + EST^{n_z}_{t_{ij}}$ (ET is the execution time of t_{ij} on n_z). $available_z$ denotes the time that n_j is available (idle state). For entry tasks having no predecessors the second argument of the EST is always the arrival time. When t_{ij} is scheduled in n_z, the $EST^{n_z}_{t_{ij}}$ and $EFT^{n_z}_{t_{ij}}$ are equal to the actual start time ($AST^{n_z}_{t_{ij}}$) and the actual finish time ($EFT^{n_z}_{t_{ij}}$), respectively.

Research Challenge: *Find a feasible mapping between tasks of w_i and nodes $\mathcal{N}$, such that the overall network overhead is minimized w.r.t. the following constraints: (i) a node cannot execute concurrently more than one tasks, and (ii) a task cannot be assigned to more than one nodes.*

IV. Probabilistic Workflow Management

Multiple workflows formulation. We consider a two level orchestration scheme as depicted by Fig. 1. In the first level, the GO, iteratively, receives the ready tasks of the envisioned workflows and assigns them to the available nodes. The GO is triggered at predefined intervals while other strategies could be also adopted (e.g., when significant updates on the present data are realized). We assume the existence of a global queue that contains the ready tasks of each workflow, i.e., tasks for which all predecessors have finished their execution. The GO assigns tasks according to a specific strategy (see section IV-1) that tries to eliminate the network overhead. In the second level, the LO is responsible to prioritize tasks within the node. Each node n_i is equipped with a local queue where ready tasks sent by the GO are placed. Tasks are considered for scheduling one by one according to a priority value (see section IV-2). The first task of the local queue is placed for execution when n_i becomes idle.

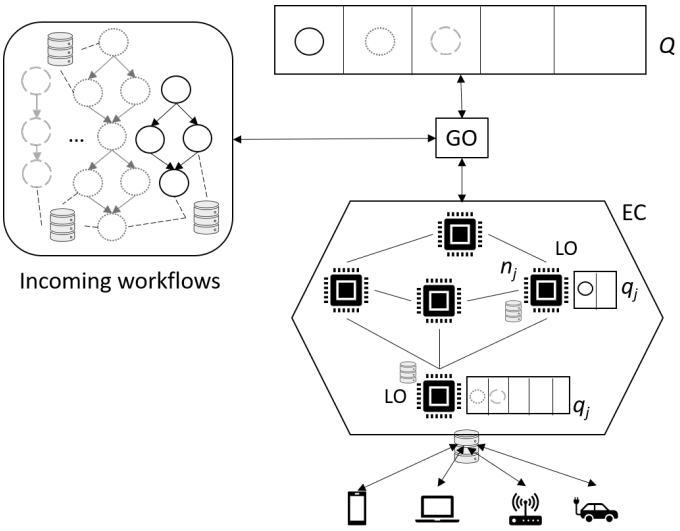


Fig. 1: System's architecture

*1) **Global orchestration**:* Our problem is to detect the appropriate node for each set of ready tasks to eliminate the need for data migration. We envision assignments that 'match' every workflow with the node that corresponds to its 'requirements': *the appropriate data and the appropriate processing speed* We propose the use of a model to detect the probability of assignment for each potential pair w_i, n_j, i.e., $P(w_i, n_j)$. From every w_i, we consider the T ready tasks $t^i_1, t^i_2, \ldots, t^i_T$ (we slightly update the above given notation to facilitate our calculations). A (sub)set of tasks demand for a specific range of data while the remaining are dependent tasks, i.e., they wait for the outcomes of their peer-tasks. Focusing on the generalized data demand of w_i which represents the demand dictated by all tasks in w_i, we get $D_{w_i} = \cup_{k=1}^{T} t^i_k$. We consider a multivariate scenario where M dimensions are adopted for every data vector. This means that w_i demands data for each dimension $m, m = 1, 2, \ldots, M$ based on a 'range' model, i.e., $D_{w_i} = \langle [min^i_1, max^i_1], [min^i_2, max^i_2], \ldots, [min^i_M, max^i_M] \rangle$. For simplicity in our calculations, we consider the middle of each interval, i.e., $D_{w_i} = \langle d^i_m \rangle$ with $d^i_m = \frac{min^i_m + max^i_m}{2}$ & $m, m = 1, 2, \ldots, M$.

On the other side, every node n_j is the owner of a dataset, i.e., a set of multivariate vectors, i.e., $\mathbf{x} = \langle x_1, x_2, \ldots, x_M \rangle$. Without loss of generality, we consider that every dimension follows a Normal distribution with mean μ_m and standard deviation σ_m. μ_m and σ_m can be updated through an incremental approach in order to save time. We discern that the assignment of a workflow in a processing node should be dictated by

the matching between D_{w_i} and $D_{n_j} = \left\langle \mu_1^j, \mu_2^j, \ldots, \mu_M^j \right\rangle$ (D_{n_j} represents the vector of means for each dimension). For simplicity in our notations, we eliminate the indexes i and j from the representation of D_{w_i} and D_{n_j}.

The aforementioned matching can be depicted by the distance between the required data and the means of local datasets. This can be estimated by the absolute value of the difference $\Omega_{ij} = |D_{w_i} - D_{n_j}| = \sum_{m=1}^{M} |d_m - \mu_m|$. We rely on the random variables X_m and Y_m depicting the data requirements and the data present at the local datasets for the mth dimension. As both variables follow a Normal distribution, the random variable $Q = X - Y$ (we omit the index m for simplicity) follows a Normal difference distribution, i.e., $Q \sim \mathcal{N}\left(\mu_X - \mu_Y, \sigma_X^2 + \sigma_Y^2\right)$ [26].

Lemma 1. *The cumulative distribution function (cdf) and the mean of the absolute difference of X & Y are given by:* $F_Z(z) = \frac{1}{2}\left[erf\left(\frac{z+(\mu_X-\mu_Y)}{\sqrt{2(\sigma_X^2+\sigma_Y^2)}}\right) + erf\left(\frac{z-(\mu_X-\mu_Y)}{\sqrt{2(\sigma_X^2+\sigma_Y^2)}}\right)\right]$ *and* $\mu_Z = \sqrt{\frac{2}{\pi}(\sigma_X^2+\sigma_Y^2)}e^{-\frac{(\mu_X-\mu_Y)^2}{2\left(\sigma_X^2+\sigma_Y^2\right)}} + (\mu_X-\mu_Y)\left(1-2\Phi\left(-\frac{\mu_X-\mu_Y}{\sqrt{\sigma_X^2+\sigma_Y^2}}\right)\right)$ *with Z being the random variable of the absolute value of Q.*

Proof. Upon Q, we can define the random variable Z which depicts the absolute difference between the required data and the available data for a specific dimension, i.e., $Z_m = |Q_m|$. Z follows a folded Normal distribution with the following probability density function (pdf) (for $z > 0$):

$$f_Z(z) = \frac{1}{\sqrt{2\pi\left(\sigma_X^2+\sigma_Y^2\right)}} \times \left[e^{-\frac{(z-(\mu_X-\mu_Y))^2}{2\left(\sigma_X^2+\sigma_Y^2\right)}} + e^{-\frac{(z+(\mu_X-\mu_Y))^2}{2\left(\sigma_X^2+\sigma_Y^2\right)}}\right] \quad (1)$$

[27] By adopting simple calculations, the cdf of Z is given by:

$$F_Z(z) = \frac{1}{2}\left[erf\left(\frac{z+(\mu_X-\mu_Y)}{\sqrt{2(\sigma_X^2+\sigma_Y^2)}}\right) + erf\left(\frac{z-(\mu_X-\mu_Y)}{\sqrt{2(\sigma_X^2+\sigma_Y^2)}}\right)\right] \quad (2)$$

Finally, the mean of Z is given by:

$$\mu_Z = \sqrt{\frac{2}{\pi}(\sigma_X^2+\sigma_Y^2)}e^{-\frac{(\mu_X-\mu_Y)^2}{2\left(\sigma_X^2+\sigma_Y^2\right)}} + (\mu_X-\mu_Y)\left(1-2\Phi\left(-\frac{\mu_X-\mu_Y}{\sqrt{\sigma_X^2+\sigma_Y^2}}\right)\right) \quad (3)$$

where $\Phi()$ is the cdf of the univariate standard normal distribution. □

Adopting the equations provided by Lemma 1, we can identify that Ω_{ij} is the summation of M folded Normal distributions, i.e., $\Omega_{ij} = \sum_{\forall m} Z_m$. In general, we want to have $\Omega \leq \theta$ where θ is a pre-defined threshold depicting the magnitude of matching between D_{w_i} & D_{n_j} Focusing on an individual dimension, we want $\beta = P\left(Z_m \leq \theta_m\right)$ with θ_m being the threshold for the specific dimension. The use of different thresholds θ_m opens up the path to adopt a strategy for handling a trade off and paying more attention on a (sub)set of dimensions demanding for a low difference.

Lemma 2. *The probability of the assignment of w_i to n_j as depicted by their data distance is given by $\Omega_{ij} = \prod_{m=1}^{M} F_Z^m\left(\theta_m\right)$*

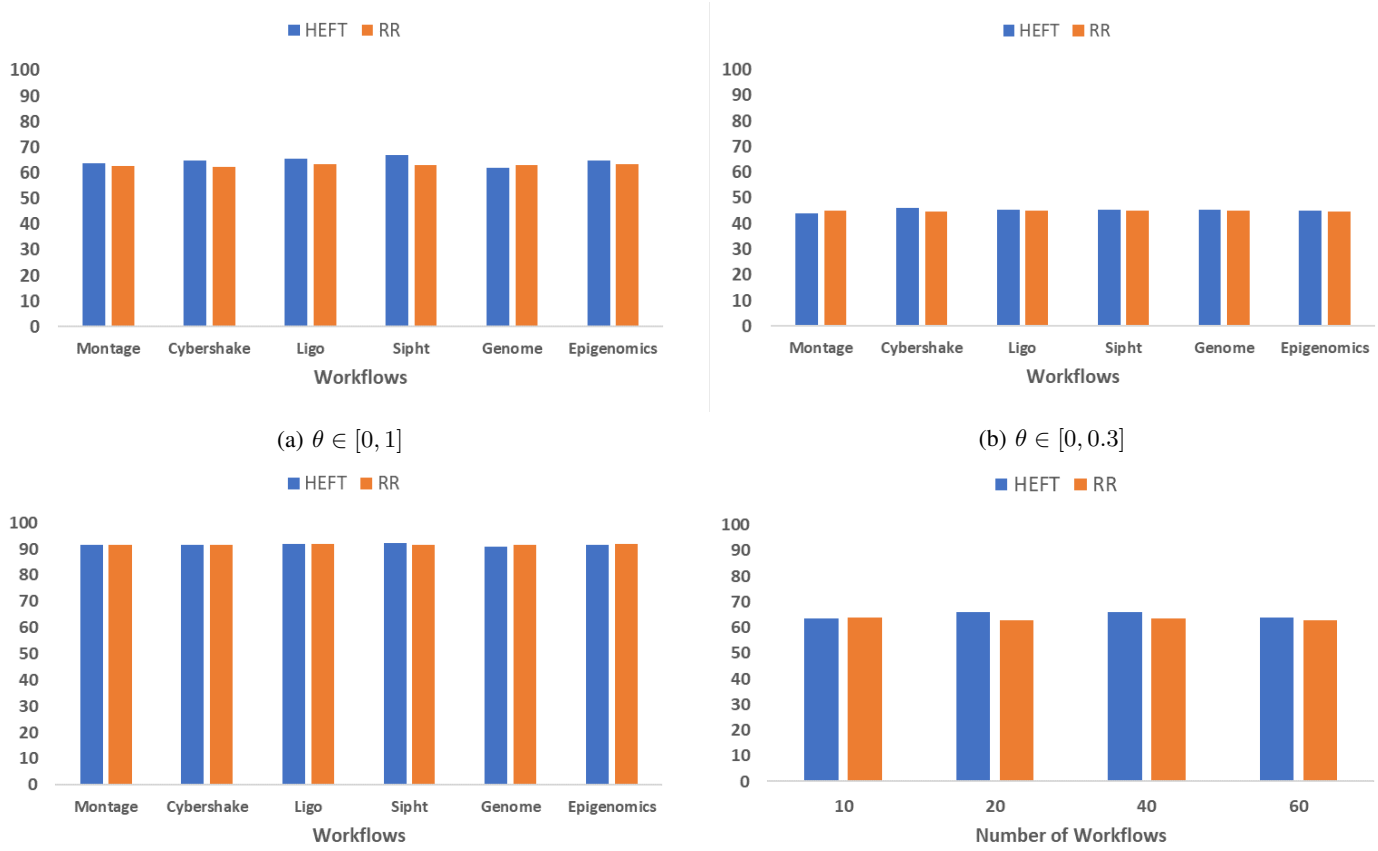
Proof. β depicts the cdf of Z_m. In our analysis, we consider independent dimensions, thus, the probability of assignment (matching) between w_i and n_j is defined by: $\Omega_{ij} = \prod_{m=1}^{M} F_Z^m\left(\theta_m\right)$ □

For each w_i, we select the node n_j that makes the following equation true: $n_j = \text{argmax}_{n_j \in \mathcal{N}} \Omega_{ij}$.

*2) **Local orchestration**:* The LO is invoked when one or more tasks arrive in a node. The LO is responsible to find the execution start time of each task and subsequently assign them to the local queue. An interesting observation is the heterogeneity of tasks, i.e., tasks have different processing demands, are bounded to different data and belong to different workflows. We depart from this non-trivial issue and split incoming tasks into two categories. The first category contains tasks that belong in the critical path of a workflow while the second category contains the rest. Tasks of the first category are prioritized for execution as their delayed execution will affect the final makespan. We should mention that when tasks arrive in a node they start requesting the appropriate data to initiate their execution. Tasks enter the local queue when they are ready for execution i.e., their EST value is equal to the current time. Tasks in both categories are scheduled based on the Shortest Job First (SJF) policy to minimize the average waiting time among them. In case new tasks arrive in the system and the local waiting queue is not empty the aforementioned policy is applied again and tasks are re-prioritized. We have to notice that if tasks are placed at a node not having similar data, tasks will demand those data from other nodes increasing the network overhead and the execution time.

V. Experimental Evaluation & Performance Assessment

Workflows, Resources & Performance Metrics. We report on the experimental evaluation of the proposed model focusing on the minimization of the need for transferring data in the network and using six (6) different workflow applications as depicted by [28], and [29]. The number of tasks, the execution time of each task as well the amount of data transferred between them is reported in a 'Directed Acyclic Graph in XML' (DAX) format. Workflows include Montage, CyberShake, LIGO, SIPHT, Genome and Epigenomics which are extensively adopted in the relevant literature. The discussed



(a) $\theta \in [0, 1]$

(b) $\theta \in [0, 0.3]$

(c) $\theta \in [0.3, 0.6]$

(d) Random Workflows, $\theta \in [0, 1]$

Fig. 2: Average network overhead improvement of PMW ϵ

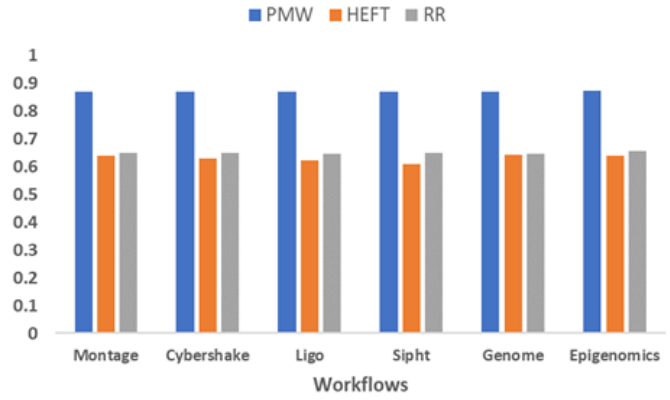
workflows 'cover' all the basic execution patterns such as pipelining, process, data aggregation, data distribution and redistribution. Each workflow is provided in different sizes with a total of 54 workflow applications being adopted in our experimentation with their characteristics given in Table I.

TABLE I: Workflow characteristics

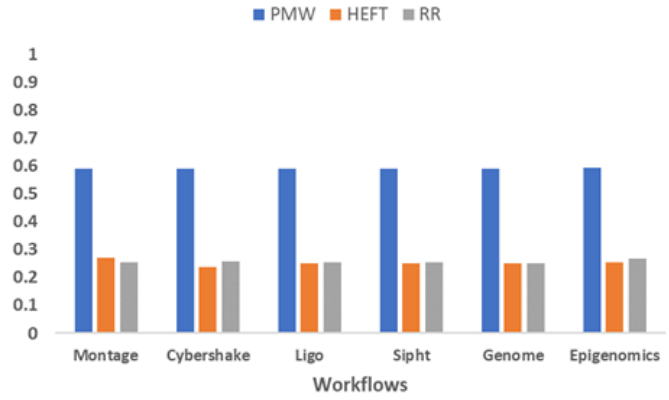
Name (Number of worflows)	Number of Tasks (min-max)	CCR (min-max)
Montage (10)	25 - 800	23.9 - 36.9
CyberShake (10)	30 - 800	206.6 - 257.8
LIGO (10)	30 - 800	0.05 - 0.26
SIPHT (10	30 - 700	0.21 - 0.24
Genome (10)	50 - 900	0.01 - 1.66
Epigenomics (4)	24 - 997	0.05 - 2.42

CCR depicts the Communication to computation value i.e., the ratio between the average communication cost and the average computation cost. CCR reveals whether a workflow is communication intensive (high value) or computation intensive (low value). For each experiment nodes processing capacity is randomly set between 1 and 16 task computational demand units considering an equal transfer rate among them. Without loss of generality, we consider that task requests for a specific range of data is realized in the unity interval for every dimension while data collected by nodes are randomly set in the unity interval too. The performance of the proposed model is evaluated by: **(i)** the total volume of data transferred through the network (V), **(ii)** the average Ω and **(iii)** the overall time (ms) required by the GO to deliver the final decisions (DT). To measure the total volume of migrated data (in case a task demands data not present in the node where the placement is realized), we assume the reward function $r(t_i, n_j)$ that returns the temporal cost of sending data demanded by t_i to n_j as follows: $r(t_i, n_j) = \sum_{k=1}^{M}(1 - F_z(\theta_i))/Comm_{w_i})$ with $Comm_{w_i}$ being the average communication cost of w_i.

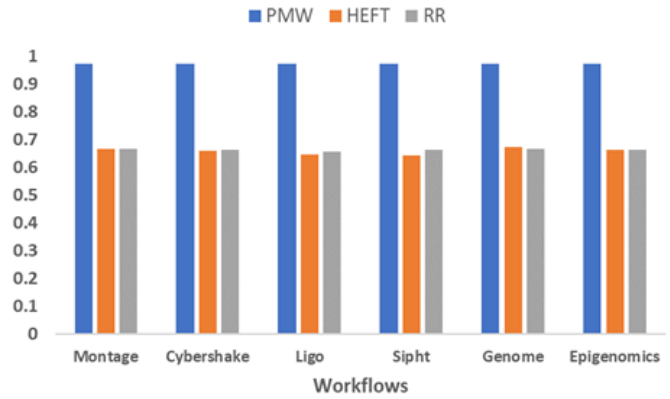
Performance Evaluation. We compare the proposed methodology, i.e., the Probabilistic Management of Workflows (PMW) against two well-known algorithms: the Round Robin (RR) and the HEFT algorithm [3]. For each task, the RR assigns tasks by selecting the processing nodes in a circular order while HEFT selects the node that results in the minimum EFT. Initially, we evaluate our model in terms of V and Ω. We consider three different realizations of θ (we consider the same threshold for all dimensions) randomly selecting it in the following intervals: (a) [0, 1], (b) [0, 0.3] (c) [0.3, 0.6]. We evaluate our scheme through an extensive set of simulations involving different number and types of workflows. The number of nodes and dimensions are set equal to 10. In Figs.



(a) $\theta \in [0, 1]$



(b) $\theta \in [0, 0.3]$



(c) $\theta \in [0.3, 0.6]$

(d) Random Workflows, $\theta \in [0, 1]$

Fig. 3: Average Ω.

2a-2c and Figs. 3a-3c, we assume that 10 workflows of the same type arrive in the system at random time units. In Figs. 2d and 3d, we increase the number of workflows and get it equal to 10, 20, 40 or 60. In each experiment, we mix the number and the types of the considered workflows.

In Fig. 2, we plot the percentage of the improvement that the PMW offers in terms of V against RR and HEFT. Actually, the plot depicts the difference between our model and the aforementioned algorithms, i.e., $\epsilon = \frac{V_{ref} - V_{PMW}}{V_{ref}}\%$ (V_{ref} is the realization of V for the discussed models RR and HEFT). We can observe a significant reduction of the data migration actions due to the appropriate placement of the desired tasks into the available nodes as concluded by the PMW. This holds true for all the experimental scenarios. When $\theta \in [0.3, 0.6]$, we can observe the maximum performance as the proposed model is more relaxed in the delivery of the matching outcome. With the term relaxed, we annotate a behaviour that leaves room for the matching process that may result a 'better' dataset compared to a 'strict' threshold that tries to seek a perfect match. In real cases, the use of a very low θ may jeopardize the quality of the matching process with the danger of not returning any outcome forcing the proposed model to rely on a completely random selection afterwards. In case where θ is realized in the unity interval (heavy fluctuations in its value are concluded in consequent iterations), we observe that the reduction of the data migration actions is less than the aforementioned scenario ($\theta \in [0.3, 0.6]$) proving our allegations.

In Fig. 3, we plot our results for Ω. Here, we can observe that the PMW achieves a higher probability of assignment that the remaining models that participate in our comparative assessment. Recall that a high Ω depicts a high matching between the data demanded by workflows and the available datasets. The interesting is that the proposed model is not affected by the type of the workflow as presented by Figs 3a, 3b, 3c. or the number of workflows as presented by Fig. 3d. We can easily discern the stability of the PMW and its attitude to select similar datasets for every workflow always targeting to minimize the network overhead.

We perform another set of experiments and evaluate the proposed model concerning the time required by the GO DT to deliver the final decision of the envisioned assignments. We adopt $M \in [10, 100]$ and $|\mathcal{N}| \in [10, 100]$. After the experimentation, we retrieve $DT \in [59.7, 500.66]$ which

leads to a throughput of 1.99 to 16.75 sets of workflows per second. We have to notice that, in this experimentation activity, we consider ten workflows per set belonging on the Genome type. Evidently, the proposed model can be efficiently incorporated into a real time decision making mechanism to support applications at the Cloud or Fog/Edge environments.

VI. Conclusions & Future Work

We propose a model that targets to assist in the scheduling of scientific workflows over distributed heterogeneous resources. We aim at combining the requirements of workflows in terms of data upon which the envisioned processing will be realized and 'legacy' approaches that target to minimize the execution time and the monetary cost. We elaborate on the use of two orchestrators, i.e., a global one that adopts a probabilistic approach to expose the distance between workflows requirements and the available datasets and a local orchestrator that performs the prioritization of tasks internally in the processing nodes. The probabilistic model assists in the assignment process while the prioritization scheme assists in the efficient execution of the assigned tasks. Our results reveal a good performance of the proposed scheme as compared with other similar efforts and baseline models. We show that the network overhead is minimized as we avoid the unnecessary data migrations due to inefficient placements. In the first places of our future research plans, one can find an intelligent technique for the management of tasks in the local orchestrator to cover the heterogeneity of tasks characteristics and their significance to the efficient conclusion of the entire workflow.

References

[1] F. Wu and et. al., "Workflow scheduling in cloud: a survey," *The Journal of Supercomputing*, vol. 71, pp. 3373–3418, 2015.

[2] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 8, p. e4041, 2017.

[3] H. Topcuoglu and et. al., "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE TPDS*, vol. 13, pp. 260–274, 2002.

[4] Y.-K. Kwok and I. Ahmad, "Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors," *IEEE TPDS*, vol. 7, pp. 506–521, 1996.

[5] P. Oikonomou, K. Kolomvatsos, N. Tziritas, G. Theodoropoulos, T. Loukopoulos, and G. Stamoulis, "Uncertainty driven workflow scheduling using unreliable cloud resources," in *19th IEEE NCA*, pp. 1–8, 2020.

[6] P. Oikonomou, K. Kolomvatsos, and T. Loukopoulos, "Resource provisioning schemes for multiple workflowscheduling with seclusion requirements," in *Panhellenic Conference of Information*, 2020.

[7] M. A. Iverson and F. Özgüner, "Hierarchical, competitive scheduling of multiple dags in a dynamic heterogeneous environment," *Distributed Systems Engineering*, vol. 6, p. 112, 1999.

[8] G. C. Sih and E. A. Lee, "A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures," *IEEE TPDS*, vol. 4, pp. 175–187, 1993.

[9] Z. Yu and W. Shi, "A planner-guided scheduling strategy for multiple workflow applications," in *2008 IEEE ICPP*, pp. 1–8, 2008.

[10] G. L. Stavrinides and H. D. Karatza, "Scheduling multiple task graphs in heterogeneous distributed real-time systems by exploiting schedule holes with bin packing techniques," *Simulation Modelling Practice and Theory*, vol. 19, pp. 540–552, 2011.

[11] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[12] T. Adam and et. al., "A comparison of list schedules for parallel processing systems," *Communications of the ACM*, vol. 17, pp. 685–690, 1974.

[13] B.-C. Cheng and et. al., "Lstf: a new scheduling policy for complex real-time tasks in multiple processor systems," *Automatica*, vol. 33, pp. 921–926, 1997.

[14] G. Stavrinides and H. Karatza, "A cost-effective and qos-aware approach to scheduling real-time workflow applications in paas and saas clouds," in *3rd IEEE ICFITC*, pp. 231–239, 2015.

[15] Y.-L. Tsai and et. al., "Scheduling multiple scientific and engineering workflows through task clustering and best-fit allocation," in *IEEE Eighth World Congress on Services*, pp. 1–8, 2012.

[16] L. F. Bittencourt and E. R. Madeira, "A performance-oriented adaptive scheduler for dependent tasks on grids," *Concurrency and Computation: Practice and Experience*, vol. 20, pp. 1029–1049, 2008.

[17] E.-K. Byun and et. al., "Cost optimized provisioning of elastic resources for application workflows," *FGCS*, vol. 27, pp. 1011–1026, 2011.

[18] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, 2011.

[19] M. Mao and M. Humphrey, "Scaling and scheduling to maximize application performance within budget constraints in cloud workflows," in *27th IEEE International Symposium on Parallel and Distributed Processing*, pp. 67–78, 2013.

[20] A. C. Zhou and B. He, "Transformation-based monetary costoptimizations for workflows in the cloud," *IEEE Transactions on Cloud Computing*, vol. 2, pp. 85–98, 2014.

[21] A. C. Zhou and et. al., "Monetary cost optimizations for hosting workflow-as-a-service in iaas clouds," *IEEE Transactions on Cloud Computing*, vol. 4, pp. 34–48, 2015.

[22] I. Petri and et. al., "Edge-cloud orchestration: Strategies for service placement and enactment," in *IEEE International Conference on Cloud Engineering*, pp. 67–75, 2019.

[23] A. Alqahtani and et. al., "Sla-aware approach for iot workflow activities placement based on collaboration between cloud and edge," in *1st Workshop on Cyber-Physical Social Systems*, 2019.

[24] M. Breitbach and et. al., "Context-aware data and task placement in edge computing environments," in *IEEE International Conference on Pervasive Computing and Communications*, pp. 1–10, 2019.

[25] Z. Chen and et. al., "Effective data placement for scientific workflows in mobile edge computing using genetic particle swarm optimization," *Concurrency and Computation: Practice and Experience*, p. e5413, 2019.

[26] E. W. Weisstein, "Normal distribution," *https://mathworld. wolfram. com/*, 2002.

[27] M. Tsagris and et. al., "On the folded normal distribution," *Mathematics*, vol. 2, pp. 12–28, 2014.

[28] G. Juve and et. al., "Characterizing and profiling scientific workflows," *FGCS*, vol. 29, no. 3, pp. 682–692, 2013.

[29] S. Bharathi and et. al., "Characterization of scientific workflows," in *3rd workshop on workflows in support of large-scale science*, pp. 1–10, 2008.