*IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)*
*e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 14, Issue 3 Ver. 3 (May-June. 2019), PP 76-81*
*www.iosrjournals.org*

# Survey on Workflow Scheduling in Cloud Environment

R Subba Rao[1], Chinmaya Ranjan Pattanaik[2]
*1(Department of Computer Science & Engineering, Gandhi Engineering College ,India)*
*2(Department of Computer Science & Engineering, Gandhi Institute For Technology ,India)*

***Abstract:*** *Workflow scheduling is one of the most challenging issues in cloud computing. Workflow is widely used paradigm in collaborative research and managing complex large scale distributed application. Various distributed environment such as cluster, grid and cloud use workflow to process complex and discrete tasks. Each task may include entering data, processing, accessing software, or storage functions. The task-resource mapping, QoS requirement, on-demand resource provisioning, performance fluctuation and failure management in workflow scheduling is considered to be an NP-hard problem. An efficient scheduling algorithms are required to select the best suitable resources for workflow execution. In this paper, a comprehensive survey of workflow scheduling strategies that have been proposed for cloud computing platforms to help researchers systematically and objectively gather and aggregate research evidences.*
***Key Word****: Cloud computing, Cost aware, Distributed system, Energy aware workflow scheduling, Workflow scheduling.*

## I. Introduction

Scientific workflow scheduling of an application on cloud computing, is a complex optimization problem which may require consideration of different scheduling criteria. Usually, the most important criteria are the expected execution time and the cost of running an application on a machine. In addition, scientific workflow applications have many computations and tasks that generate many intermediate datasets with large size. There exist dependencies among the intermediate datasets. So, the scheduler should also take care of precedence constraints between the set of tasks. In its most general form, the problem of tasks scheduling of a graph onto a set of different resources is an NP-Complete problem [1]. As a result, over several years, a number of heuristic algorithms suitable for workflow scheduling on heterogeneous resources have been suggested [2] that attempt to strike a good balance between running time, complexity and schedule quality [3], but still a lot of work needs to be done for making scheduling in clouds more effective.

The aim of this paper is to provide a comprehensive survey on workflow scheduling techniques in different existing distributed environments. The first section gives an introduction to workflows and workflow scheduling. Then, a survey based on various scheduling objectives are discussed and a comparative study of various algorithms are also done.

### 1.1. Workflow Modelling

A wide range of scientific applications in distributed systems [4] can be modelled by a workflow which is a directed acyclic graph (DAG) G = (T,E) where T is a set of nodes or tasks (t1,t2,t3……tn} and E is the set of directed edges {eij | (ti,tj) ∈ E} representing the dependencies between the tasks. Each task is a workflow task with an associated computation workload wli . Each edge eij represents ti as the parent task of tj and tj is said to be the child task of ti. Only after the complete execution of the parent task, a child task can be executed. If there is data transmission from ti to tj, the tj can start only after all the data from ti has been received. A task which does not have parent task is called an entry task whereas a task which does not have child task is called an exit task. Generally, there are two types of workflow which are simple and scientific workflows. Figure 1 indicates a simple workflow's DAG. It shows a 7-node DAG, where node T1 is the first task to be executed and it is the entry task node ,nodes T3, T4 and T5 can only start their execution after task T1 finishes and sends the data, and node T6 can only start their execution after task T3 finishes and sends the data. The node T2 can be executed only after nodes T4 and T5 completes execution. The last task that is the last task, node T7 can only start its execution after all its parent tasks finish and send their data. Nodes in the DAG are labeled with their computation cost (number of instructions, while edges are labeled with their communication cost (bytes to transmit).

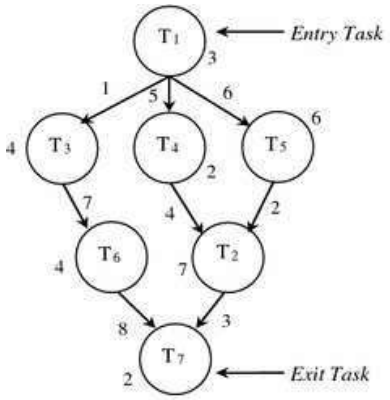
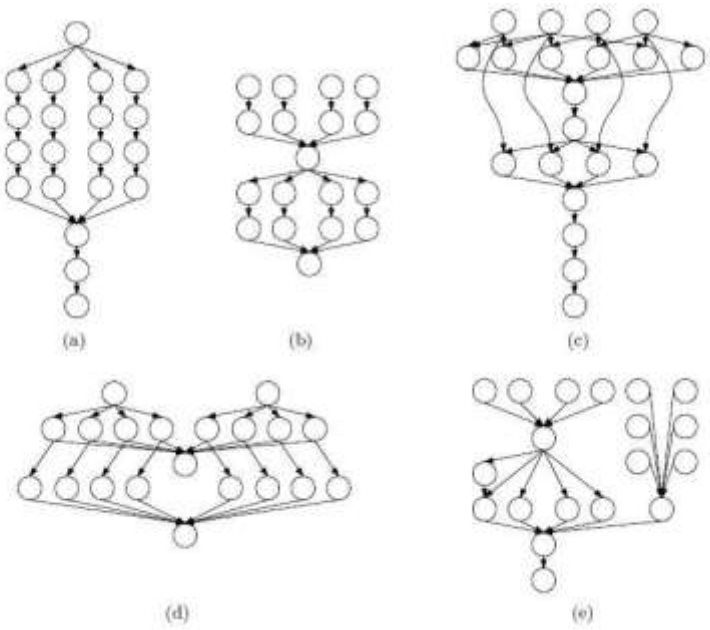



**Figure 1** DAG Workflow graph

Also, there are numerous scientific Workflows as shown in the figure 2. There are five real workflow applications that are used in the scientific domains, namely, Epigenomics [5], LIGO Inspiral Analysis Workflow [6], Montage [7], CyberShake [8], and SIPHT [9]. The Epigenomics workflow created by the USC Epigenome Center and the Pegasus framework is used to automate the different operations in genome sequence processing. LIGO's Inspiral Analysis workflow is used to create and analyse gravitational waveforms from data gathered during the coalescing of compact binary systems. The Montage application created by NASA/IPAC closes together multiple input images to form custom mosaics of the sky. The CyberShake workflow is used by the Southern California Earthquake Centre to distinguish earthquake threatening a region. The SIPHT workflow, from the bioinformatics project at Harvard, is used to automate the search for small untranslated RNAs (sRNAs) for bacterial replicons in the NCBI database

In workflow scheduling, clients submit their tasks to a scheduling server, which works as an intermediate between the cloud users and cloud provider, through a client terminal which is also in charge of initializing the tasks and generating task information table including task number, storage space required, task type, etc. Then, the scheduler takes tasks from users and allocates them to appropriate resources for task execution according to a scheduling algorithm. When tasks execution are finished, the computational nodes return the results to the scheduling server, and the data including computing result and operation information will be sent back to the cloud client.




## II. Classification of the Workflow Scheduling

The main objective of workflow scheduling is to achieve the expected goal by dispatching tasks to appropriate resource for execution. Currently, the common objectives for workflow scheduling schemes include makespan, cost, deadline , energy, QoS, load balancing ,security and fault-tolerance.

2.1. Makespan Based Workflow Scheduling

A schedule for a workflow represents the assignment of tasks to resources. It is defined as a S =< R, M,makespan >, where R = {r1, r2, . . . , r|R|} is a set of used resources, and M consists of all task-resource mappings. The makespan is the overall schedule of the workflow. GA-PSO algorithm [4] reduces the makespan through a fair utilization of the slow VMs instead of overloading the fast VMs and slowing down the overall execution of the tasks. The design of the GA-PSO [10] algorithm uses the standard deviation to select the best solution that keeps the variance of the distributed load, over the VMs, as low as possible taking into account the size of the tasks and the speed of each VM during the distribution of the tasks. The GA-PSO algorithm yields an optimal solution of the workflow task scheduling in terms of makespan compared with GA[11], PSO[12], HSGA[13], and WSGA[14] algorithms by 16%, 4%, 11%, and 5%, respectively. This algorithm can be improved through handling dynamic workflow in a heterogeneous environment. Authors in [15], have proposed the Intelligent Water Drops (IWD) algorithm which is a new meta-heuristics, is customized for solving job-shop scheduling problems in cloud computing environment. To increase the diversity of the solution space as well as the solution quality, five schemes are proposed. In addition, to improve the original IWD algorithm, an improved algorithm named the Enhanced IWD is proposed. The optimization objective is the makespan of the schedule. Authors demonstrated that the EIWD algorithm can find better solutions for the standard benchmark instances than the existing makespan based techniques. Cloud computing raises new challenges to efficiently allocate resources for the workflow application and also to meet the user's quality of service requirements. To deal with these challenges, Lu et al. [16] propose an adaptive penalty function for the strict constraints compared with other genetic algorithms. They used co evolutionary approach to adjust the cross-over and mutation probability. This helps in accelerating the convergence and prevents prematurity. This algorithm is compared with Random, HEFT, PSO and Genetic algorithms [30] in a WorkflowSim simulator on four representative scientific workflows. Experiment results show that the proposed algorithm produced results better than PSO, GA, HEFT and Random Scheduling algorithms in the criterion of both the deadline-constraint and the total execution cost.

**2.2. Cost Based Workflow Scheduling**

In Cloud based workflow scheduling the aim is to finish a workflow as fast as possible at given budget. The proposed CEAS [11] algorithm consists of five sub-algorithms. The five sub algorithms used are VM selection algorithm, two tasks merging methods, reuse the idle VM instances, and the task slacking algorithm. The sequence tasks merging, parallel tasks merging and VM reuse algorithms can reduce the cost of workflow efficiently. The five sub algorithms can considerable save energy also. Experiment results shows that CEAS algorithm outperforms existing well-known approaches in both cost and energy consumption. A Hybrid Cloud Optimized Cost (HCOC) scheduling algorithm Bittencourt et al. [12] was proposed to schedule workflow in hybrid environment. This algorithm combines a private cloud with pubic cloud to reduce cost and increase the speed of execution of workflows. The workflows are scheduled based on HEFT [13] algorithm that schedules workflows in hybrid clouds by first attempting costless local scheduling. To execute a workflow within a given execution time, the local scheduling algorithm leases resources from the public cloud to provide sufficient processing power. But this algorithm does not deal with multiple workflows and estimate the available bandwidth between two public clouds. Such estimative is essential for the scheduling algorithm to decide when dependent tasks could be put in different public clouds.

A trust service-oriented workflow scheduling algorithm [14] proposed by Tan combines direct trust and recommendation trust. The weight of cost is incrementally adjusted until the execution time of all tasks satisfies the deadline. It is possible to find an optimum solution with the deadline constraint by adjusting the weights of time and cost effectively and feasibly. In addition, they provide balance policies to enable users to balance different requirements, including time, cost, and trust. A case study was conducted to illustrate the value of the proposed technique. A cost optimization algorithm [15] for scheduling scientific workflows on IaaS (Infrastructure as a Service) clouds such as Amazon EC2. Applications are scientific workflows modelled as DAGs as in the Pegasus Workflow Management System. They assume that tasks in the workflows are grouped into levels of identical tasks based on mathematical programming languages (AMPL and CMPL). A mixed integer nonlinear programming problem to solve the scheduling of large scale scientific applications on hybrid clouds, where the optimization objective is the total cost, with a deadline constraint

**2.3. Deadline Constrained Workflow Scheduling**

The proposed CCA [16] algorithm will not violate the deadline while reducing the cost. This method uses flexible scoring approach and consider different criteria like leasing cost, makespan and resource utilization to combine the available cluster in the workflow. The scoring function is adjusted to meet the deadline. Another deadline-constrained workflow scheduling algorithm uses minimum critical path (MCP). It only chooses cheaper resources for non-critical tasks, under the condition that execution of critical tasks is not influenced.

Hence, the cost is reduced without increasing the makespan. If the makespan is smaller than deadline, there is potential to further reduce cost by delaying makespan to the deadline. A general deadline distribution heuristic consists of three phases: task partition, deadline assignment, and resource selection. DTL [17], DBL [18], and DET [19] are three heuristics implementing the deadline distribution strategy. DTL and DBL partition the tasks into different levels in accordance of their parallel or synchronization properties. The difference between DTL and DBL is partition direction. DTL partitions tasks in top-bottom direction while DBL in the opposite direction. All tasks in the same level have the same subdeadline. At the deadline assignment phase, DTL divides the overall deadline over task partitions in proportion to their minimum execution time (MET). While DBL divides the deadline by amortizing float time uniformly to all levels. For the start time of a task, DBL does not require tasks of the same level to start at the same time as DTL. DBL relaxes this limit by starting from the finish time of critical parent. As opposed to leveling-based deadline distribution strategy, the DET algorithm partitions all tasks into different paths of an Early Tree. The whole deadline is divided into time windows for critical tasks, which can be applied to all feasible deadlines. An iterative algorithm is proposed to determine the time windows for non-critical tasks while keeping the precedence constraints among tasks. At resource selection phase, DBL selects the cheapest resource while guaranteeing that the task execution can complete within its sub-deadline. DTL adopts Markov Decision Process (MDP) and DET adopts dynamic programming method to search local optimal resources for tasks. Abrishami et al. [20] presented a Partial Critical Path (PCP) based task partition method evolved from their previously PCP algorithm [21] for grid environment. They proposed two algorithms of IC-PCP and IC-PCPD2 for scheduling scientificworkflow in cloud. IC-PCP distributes deadline to partitions, while IC-PCPD2 further assigns sub-deadline to each task in proportion to their METs. And resources are selected for partitions and tasks, respectively.

**2.4. Load-Aware Workflow Scheduling**

Authors in [22] examined the reasons that cause Runtime Imbalance and Dependency Imbalance in task clustering. Then, horizontal and vertical (clustering) balancing methods are proposed to address the load balance problem when performing task clustering for five widely used scientific workflows. Task clustering is a runtime process, combining multiple short execution time tasks into a single job, using this process the scheduling overhead are minimized and the improvement in runtime performance. Finally, we analyze the relationship between these metric values and the performance of proposed task balancing methods. Simulation results show that task clustering methods give a considerable progress over baseline execution in terms of load balancing among the set of tasks.

In this paper [23], authors propose a load-balanced scheduling technique for workflow applications in a cloud environment. The proposed algorithm works in two phases. They calculated priorities of all the tasks in the first phase. Then, they select virtual machines and schedule tasks in the second phase. The overall load to be executed immediately after the execution of current task is also taken into consideration by this technique. The simulated results are compared withthe benchmark scheduling heuristic named as heterogeneous earliest finish time (HEFT) and a variation of the proposed technique. The results show that the proposed approach remarkably display the performance metrics i.e., minimization in makespan and maximization in average cloud utilization. A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems was proposed by Israel Casas et al. in the paper [24]. The authors consider that incrementing number of resources does not guarantee execution time reduction and proposed BaRRS algorithm that splits scientific workflows into multiple sub-workflows to balance system utilization via parallelization. BaRRS analyzes the key application features (e.g., task execution times, dependency patterns and file sizes) of scientific workflows for adapting existing data reuse and replication techniques to cloud systems. They conclude that the optimal number of virtual machines depends on workflow characteristics. Experiments prove its superior performance compared to a state-of-the-art scheduling techniques.

**2.5. Reliability-Aware Workflow Scheduling**

Reliability in cloud computing is how consistently a cloud computing system is able to provide its services without interruption and failure. Failures are inevitable in such large complex distributed systems. It is also well studied that cloud resources experience fluctuations in the delivered performance. These challenges make fault tolerance an important criterion in workflow scheduling. Authors in [25] propose an adaptive, just-in-time scheduling algorithm for scientific workflows. They used resubmission strategy to find another suitable process unit to re-execute the task after a fault happened. This algorithm uses both spot and on-demand instances to reduce cost and provide fault tolerance. To model the failure characteristics of a cloud environment, authors in [26] developed a Monte Carlo Failure Estimation (MCFE) algorithm that considers Weibull distributed failures in cloud. Monte Carlo method can correctly model a complex system and give results that are near to complex system operations.

This approach can also minimize computation time by using divide and merge pattern for parallelization. Authors proposed Failure-Aware Resource Scheduling (FARS) algorithm that considers the reliability of task execution while assigning tasks in a workflow application to virtual machines. FARS Algorithm is an extension of the famous HEFT algorithm. The proposed algorithm is compared with HEFT using cloudsim toolkit using makespan as their performance metrics. Results show that FARS algorithm performed better than HEFT. Reliability is widely identified as an increasingly relevant issue in heterogeneous service-oriented systems because processor failure affects the quality of service to users. Replication-based fault-tolerance is a common approach to satisfy application's reliability requirement. In [27], authors dealt with this issue and proposed the heuristic replication for redundancy minimization (HRRM) method, which exhibited significant improvement in resource cost reduction and satisfaction of application's reliability requirement with low time complexity. Experimental results on real parallel applications verify that (HRRM) can generate least redundancy.

### 2.6. Energy-Aware Workflow Scheduling

In this paper [28], authors proposed a new scheduling approach named Pre Ant Policy that consists of a prediction model based on fractal mathematics and a scheduler on the basis of an improved ant colony algorithm. This efficient prediction model is developed to assist the algorithm that decides to turn on/off hosts. It helps to avoid the performance and energy loss, which is triggered by instantaneous peak loads on account of scheduling, and the scheduler is responsible for resource scheduling while minimizing energy consumption under the premise of guaranteeing the Quality-of-Service (QoS). Experimental results demonstrate that the proposed approach exhibits excellent energy efficiency and resource utilization. Traditional research in workflow scheduling mainly focuses on the optimization constrained by time or cost without paying attention to energy consumption. Through this way, Yassa et al. [29] formalized this problem as a multi-objective optimization problem, and solved it using meta-heuristic algorithms of genetic algorithm (GA) and particle swarm optimization (PSO) algorithm separately. This approach allows processors to operate in different voltage supply levels by sacrificing clock frequencies. A compromise between the quality of schedules and energy is involved by this multiple voltage. Simulation results highlight the robust performance of the proposed technique.

## III. Conclusion

In recent years, workflow scheduling has evolved to become a critical factor that can significantly affect the performance of cloud computing environment. This crucial issue is addressed by many researchers. Hence, in this paper we performed a review on existing literature related to this topic. A description and discussion of various algorithms is also included and it aims to provide further details and understanding of prominent techniques as well as further insight into the field's future directions. Through extensive literature survey, it has been found that there are many algorithms for workflow scheduling, and these algorithms somehow differ in scheduling factors and parameters. We discuss these factors in general with their associated challenges and issues namely, resources utilization, total execution time, energy efficiency, etc. It has also been analysed that workflow scheduling is NP complete problem therefore it is impossible to generate an optimal solution within polynomial time and algorithms focus on generating approximate or near-optimal solutions. From the literature reviewed, it is clear that a lot of work has already been in the area of workflow scheduling but still there are many areas which require further attention. For instance, in QoS challenges, researchers have paid less attention on availability and security aspects than others such as makespan. However, consideration of aforementioned aspects can further improve the robustness and flexibility of workflow scheduling algorithms.

## References

[1]. Pinedo ML. Scheduling: theory, algorithms, and systems. Springer; 2016 Feb 10.
[2]. Bittencourt LF, Sakellariou R, Madeira ER. Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm. In IEEE Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on 2010 Feb 17 (pp. 27-34).
[3]. Kwok, Y.-K., Ahmad, I.: Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surv. 31(4), 406–471 (1999)
[4]. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: an overview of workflow system features and capabilities. Future Gen. Comput. Syst. 25, 528–540 (2009)
[5]. Deelman E, Vahi K, Juve G, Rynge M, Callaghan S, Maechling PJ, Mayani R, Chen W, da Silva RF, Livny M, Wenger K. Pegasus, a workflow management system for science automation. Future Generation Computer Systems. 2015 May 31;46:17-35.
[6]. Brown DA, Brady PR, Dietz A, Cao J, Johnson B, McNabb J. A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis. InWorkflows for e-Science 2007 (pp. 39-59). Springer, London.
[7]. Jacob JC, Katz DS, Prince T, Berriman BG, Good JC, Laity AC, Deelman E, Singh G, Su MH. The montage architecture for grid-enabled science processing of large, distributed datasets.
[8]. H. Magistrale, S. Day, R. W. Clayton, and R. Graves, "The SCEC southern California reference three-dimensional seismic velocity model version 2," Bulletin of the Seismological Society of America, vol. 90, no. 6, pp. S65–S76, 2000.

[9]. J. Livny, H. Teonadi, M. Livny, and M. K. Waldor, "Highthroughput, kingdom-wide prediction and annotation of bacterial non-coding RNAs," PLoS ONE, vol. 3, no. 9, Article ID e3197, 2008.
[10]. Manasrah, Ahmad M., and Hanan Ba Ali. "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing." Wireless Communications and Mobile Computing2018 (2018).
[11]. Li Z, Ge J, Hu H, Song W, Hu H, Luo B. Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. IEEE Transactions on Services Computing. 2018 Jul 1;11(4):713-26.
[12]. Bittencourt LF, Madeira ER. HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. Journal of Internet Services and Applications. 2011 Dec 1;2(3):207-27.
[13]. Samadi Y, Zbakh M, Tadonki C. E-HEFT: Enhancement Heterogeneous Earliest Finish Time algorithm for Task Scheduling based on Load Balancing in Cloud Computing. In 2018 International Conference on High Performance Computing & Simulation (HPCS) 2018 Jul 16 (pp. 601-609).
[14]. Tan W, Sun Y, Li LX, Lu G, Wang T. A trust service-oriented scheduling model for workflow applications in cloud computing. IEEE Systems Journal. 2014 Sep;8(3):868-78.
[15]. Malawski M, Figiela K, Bubak M, Deelman E, Nabrzyski J. Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization. Scientific Programming. 2015 Jan 1;2015:5.
[16]. Deldari A, Naghibzadeh M, Abrishami S. CCA: a deadline-constrained workflow scheduling algorithm for multicore resources on the cloud. The journal of Supercomputing. 2017 Feb 1;73(2):756-81.
[17]. Yu J, Buyya R, Tham CK. Cost-based scheduling of scientific workflow applications on utility grids. Ine-Science and Grid Computing, 2005. First International Conference on 2005 Jul 1 (pp. 8-pp). Ieee.
[18]. Yuan Y, Li X, Wang Q, Zhang Y. Bottom level based heuristic for workflow scheduling in grids. Chinese Journal of Computers-Chinese Edition-. 2008;31(2):282.
[19]. Yuan Y, Li X, Wang Q, Zhu X. Deadline division-based heuristic for cost optimization in workflow scheduling. Information Sciences. 2009 Jul 4;179(15):2562-75.
[20]. Abrishami S, Naghibzadeh M, Epema DH. Cost-driven scheduling of grid workflows using partial critical paths. IEEE Transactions on Parallel and Distributed Systems. 2012 Aug;23(8):1400-14.
[21]. Abrishami S, Naghibzadeh M, Epema DH. Cost-driven scheduling of grid workflows using partial critical paths. IEEE Transactions on Parallel and Distributed Systems. 2012 Aug;23(8):1400-14.