An Improved Cost effective Workflow Deadline Aware TOF Work Flow Scheduling on Hybrid Cloud using TOF Montage Framework

Kumaresan.K[1], Prakash.K.P[2], Sharan.N[3], Venkadesh.T[4],Navaneethamahavishnu.A[5]

[1]Assistant Professor, Department of Computer Science and Engineering, KSRCE, TamilNadu, India

[2, 3, 4, 5] Student, Department of Computer Science and Engineering, KSRCE, TamilNadu, India.

Email: kkumaresanphd@gmail.com[1], kpprakash1613062@gmail.com[2], sharanstark2503@gmail.com[3], venkatdeshharan@gmail.com[4], navaneethavishnu322@gmail.com[5]

Abstract - Many applications running in cloud computing environment are workflow applications which contain large number of precedence specific tasks and so require proper schedule in order to complete successfully. Efficient scheduling of workflow applications is a challenging task. In workflows, the uncertainties like ‘uncertain data transfer time’ among dependent tasks and the uncertain task execution time, if ignored may lead to deadline violation. The proposed TOF Work Flow scheduling algorithms so far unconcerned these uncertainties. This paper presents an improved uncertainty aware TOF Work Flow Scheduling algorithm abbreviated as i-TOF that considers the uncertainties of scheduling workflows such as the uncertain running time of tasks in distributed environment when focused upon gave the superior outcome in way of cost and resource utilization, for DAG when compared with the original algorithm. The compared task scheduling algorithms are implemented in Workflowsim.

Keywords - Scheduling, Tasks, Uncertainties, Workflow

I.INTRODUCTION

Fields like biology, chemistry, physics, finance, mathematics etc come under scientific computing areas. The scientific applications are required to be run timely and speedily [2]. Scientific applications produce a number of workflow tasks. Such tasks which are related (such applications are called workflow applications) have to be appropriately sorted. In general, a DAG (Dynamic acyclic graphs) is used to represent a workflow. Figure 1 shows the example of a DAG.

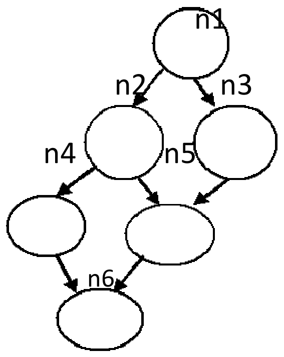


Fig. 1 An example of DAG

TOF Work Flow Scheduling means planning the allocation of workflow tasks to virtual machines based on the scheduling algorithms [3]. Other than makespan being the primary objective for effectiveness in scheduling, cost factor also matters in mutual benefit of both the users and the cloud providers. This paper compares the uncertainty aware algorithm with traditional algorithm on the basis of cost. All these factors only matter when a particular workflow completes on time. So the purpose of the algorithm presented here, is to complete the subsequent dependent tasks on time along with optimizing the cost. The broker or scheduler decides the virtual machines to which the tasks are to be allocated on the basis of specified QOS constraints like deadline and cost. The task scheduling in such applications is a complex process since the processing of all the tasks depends on the output data of their predecessor tasks. There are many historical approaches in workflow scheduling. Their one of the most prominent weakness is that they all assumed that the tasks running time and data transfer time are pre-known. However, in actual cloud systems the execution time of the tasks in any workflow cannot be predetermined. In this paper we compare the investigated Improved Uncertainty Aware Online TOF Work Flow Scheduling Algorithm abbreviated as i- TOF with the original TOF and a traditional algorithm FCFS.

II.MOTIVATION

The uncertain factors in dependent tasks' scheduling, the 'uncertain task execution time', the 'uncertain data transfer time' among tasks and the 'sudden arrival' of new workflows may result in non optimized allocation. Hence these uncertainties result in over utilization of resources which can in turn increase the users' expense. Keeping too short time for tasks may not be right, as the actual execution time may be longer, it can vary according to the network bandwidth(to transfer data among tasks) or due to different capabilities of virtual machines in distributed environment. This further can postpone the waiting tasks and the successor tasks and in turn result in deadline invasion. Whereas, reserving large amount of time too may be inappropriate since its real running time may be too less based on resources availability. Hence, running time of jobs and data transfer time among the tasks are 'uncertainties'. To solve the above problem, the sub problems are formulated in investigated paper: (1) how to diminish these uncertainties to give baseline schedules, (2) how to diminish the free time slots of the instances to reduce renting costs, while meeting deadlines. The paper ahead is in the sequence: Section 2 is dedicated to Related Work. Section 3 gives the proposed methodology. Section 4 explains the implementation and experimental results. As such experimental settings are introduced and performance results of algorithms are analyzed. Finally, in Section 5 paper is concluded with future work.

III.RELATED WORK

We start our analyses by the traditional algorithm FCFS. First Come First Serve (FCFS) algorithm is usually considered for parallel processing. It selects the instances having least waiting tasks. The disadvantage of FCFS is that it is non- preemptive. In non-preemptive scheduling the processing of tasks occurs according to their incoming order. The tasks having long makespan may finish before shorter tasks [4].Min-Min algorithm arranges the tasks according to their length. The shorter length tasks are first given to the machines having least expected completion time. Drawback of this procedure is that it chooses small tasks to be executed firstly, this in turn delays the longer tasks [5]. Max-Min algorithm is like Min-Min, but it arranges the tasks according in descending order of their length. Drawback of the algorithm is that it firstly chooses large tasks to be executed, this in turn delays the shorter tasks. Both max-min and min-min keep updating the available time of machine, i.e. suppose task 1 takes 80 seconds to complete , the execution time of next tasks are expected to be 80 seconds [6].Priority scheduling algorithm considers the priority of jobs for scheduling. It is based on multiple criteria decision making model. The list of the jobs is sorted on basis of priority. The task with greatest priority is chosen and it assigns it to resource that has minimum expected completion time. Drawback of the algorithm low priority processes may wait long time [7].Round Robin algorithm applies the time slicing technique to schedule the tasks. Each task is given a pre- decided time slice. After it operates for specified time slot, the next waiting task on that particular instance is allocated the memory, CPU and storage of that instance, again for that fixed time slot [8]. The next subsection tells about the uncertainty- aware architecture to execute workflows in cloud service environment.

There has been rapid research in workflow scheduling. The latest research in the field includes lot of successful algorithms described ahead. Lagrange relaxation aggregated cost algorithm [20] found out scheduling decision arrangement of every task to reduce the utilization of power on the mobile while focusing on meeting deadline. Design of this TOF Work Flow Scheduling algorithm saved energy utilization when differentiated to local execution and came out to be more optimal than remote execution in terms of flexibility. Numerous workflow designed applications are stored in cloud. The proposed algorithm [21] Extended dynamic constraint algorithm (add-on of multiple choice knapsack problem- MCKP) was compared with prevailing scheduling algorithm - Extended Dynamic Constraint Algorithm (EDCA). It guaranteed that monetary cost is optimized along with secure and reliable operation. It reduced 25% failures while generating the cheapest solutions among three algorithms.

Resource allocation for TOF Work Flow Scheduling always persists as a problem. Next is the study of a novel hybrid algorithm CR-AC. It came on combining the chemical reaction optimization and ant colony optimization algorithms which were proposed [22] to optimize the workflow scheduling. When compared with traditional CRO, ACO and recent PSO and CEGA; it is observed that the new algorithm gives finer results in terms of makespan and cost of planning the schedule of three workflows, on a number of machines. It achieves a high-standard optimal schedule with negligible cost meeting the deadline constraint. Outperforms given algorithms in all cases (different tasks on different no of VMs). TOF Work Flow Scheduling is a main issue in cloud computing, execution of inter dependent tasks take place along with considering Quality of Service (QOS) requirements. Quality of Service (QOS) aware TOF Work Flow Scheduling algorithm [23] compared 20-30 of heuristic, meta-heuristic, and hybrid algorithms with their numerous QOS constraints. It was concluded that most algorithms took makespan and cost as the reduction motives, so future work in TOF Work Flow Scheduling must consider fault tolerance along with load balancing, workflows security as well as protection of cloud recourses. There are many TOF Work Flow Scheduling algorithms designed, neither of which considered uncertainties associated with workflows. The uncertain execution time on a machine, and the random uncertain arrival of workflows resulted in the coming of uncertainty aware Online Scheduling Algorithm [1] abbreviated as TOF. Being aware of uncertainties, this algorithm optimizes service renting cost, resource utilization, schedule deviation and resource utilization fairness.

IV. PROPOSED METHODOLOGY

A. Service Instance Modelling:

Differently parameterised service instances are provided in cloud platform [9], [10], [11]. Let us use the symbol suk as the type u of k- th service instance; m denotes the number of obtainable service types, u: {1,2,…,m} refers service type index. Distinct types of service instances in {1,2,….,m} refer to different configurations and prices. Price (u) symbolize for cost of service type 'u' and the configurations like CPU, memory sizes and network bandwidth vary. The instances in cloud platform are priced by integer time units. The instances can be released at any point of time [12]. In this paper, it is supposed that the service instances will only be released if they have processed the entire allocated task and transferring of data.

B. System Model and Architecture:

To diminish the effect of uncertainties introduced, the explored approach, the uncertainty aware algorithm develops a novel architecture for executing workflows in cloud service environment depicted in Figure 2. The service platform can be split into three components, i.e., cloud clients, a stock of instances and a scheduler. The buyers can give in the complex applications to the cloud platform no matter when. The cloud platforms then supply a pool of service instances which can be increased or reduced in number on the basis of demand, dynamically. The scheduler acts as the mediator which maps the incoming tasks from user area to service instances considering the predefined objectives and meeting the constraints specified in applications and by platform providers.

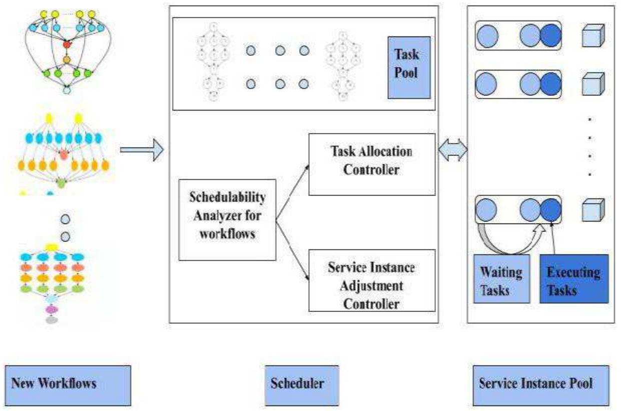

Task Pool
Task Allocation Controller
Schedulability Analyzer for workflows
Service Instance Adjustment Controller
Waiting Tasks
Executing Tasks
New Workflows
Scheduler
Service Instance Pool

Fig. 2 The uncertainty -Aware Scheduling Architecture

The scheduling strategy in this work of the scheduler is as follows. The units of scheduler are: task pool (TP), workflows schedulability analyzer, task allocation controller, and instance adjustment controller. The task pool contains the waiting workflow tasks, these waiting tasks are allocated to service instances by schedulability analyzer, it also makes the plan for service instances adjustment. This plan consists of when to lease new instances of distinct types, and the execution of this plan is done by adjustment controller. Furthermore, the task controller performs the foremost responsibility of allocating the tasks to selected service instances according to the specified plan. The next sub-section tells the algorithm to implement this procedure.

C. Algorithm Description:

For the above depicted scheduling architecture, the implemented working process can be explained in following way-

- When new workflows enter, the ranking of tasks is the prime step in this algorithm. Then, immediately, all the ready tasks are given to the machines as running tasks or the waiting tasks. If the ready for use instances are deficient, then new service instances are released. Furthermore, the remaining tasks, non-ready tasks are placed in the task pool.
- Case two can be the second uncertainty considered, i.e., when the workflow arrives even when the machines are busy. As soon as this task is completed, the waiting task on that instance is executed provided, all data from its predecessor task is received. Furthermore, the successor tasks of the completed task become ready. So the next appropriate action is to allocate these tasks that are ready to the instances instantly. In the same way, if the ready for use service instances are deficient, then new service instances are released.

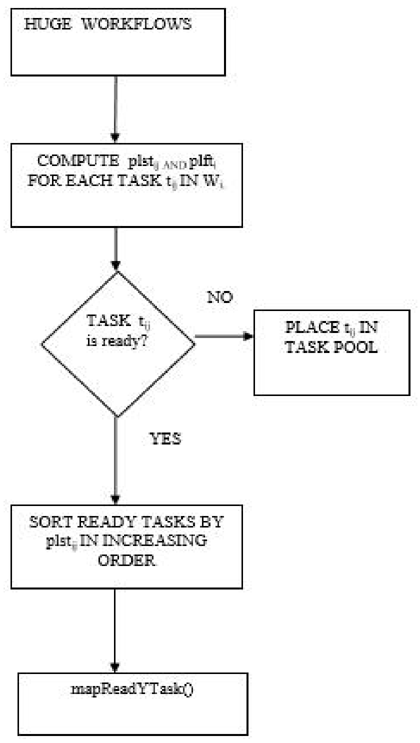
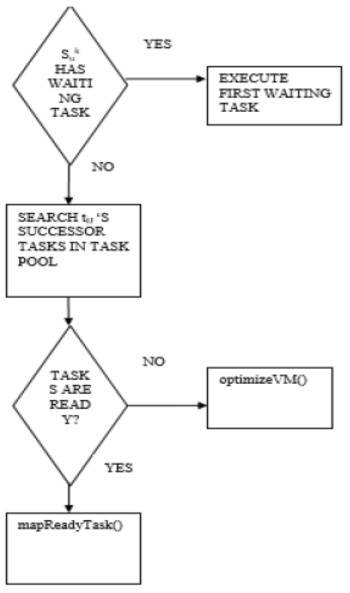
The interesting feature of this setup is that only the ready tasks can be mapped or can wait directly on the service instances, and the waiting tasks are kept in the task pool.In order to bring the above strategy into existence, these rules are followed.

- Rule 1. Only one workflow task can run on a service instance at a time.
- Rule 2. Tasks waiting on the service instance can run as soon as they receive the data from their predecessor tasks.
- Rule 3. Only when the tasks allocated and the data transfer for all the dependent tasks executed by a machine are fully completed, then only the instances are freed.

Figures 3 and 4 diagrammatically explains the proposed approach through flowchart. With contrast to the traditional scheduling schemes [9], [12], [13], on the arrival of new workflows (in Figure 3), the ranking of tasks is the prime step in this algorithm. Then, immediately, all the ready tasks are allocated to the machines as executing tasks or the waiting tasks and all the waiting tasks are placed in the task pool. See Figure 3 and Figure 4 for understanding this process diagrammatically. The other case, in Figure 4, is when an instance completes a workflow task. As soon as this task is completed, the waiting task on that instance is executed provided, all data from its predecessor task is received. Furthermore, the successor tasks of the completed task become ready and are sorted according to their $plst_{ij}$. Then, all these ready tasks are allocated to service instances instantly and are removed from the task pool. The mapping is done by function mapReadyTask() guaranteeing that the predicted finish time (i.e., $plft_{ij}$) of the workflow task is lower than its deadline. For the better optimization of cost and resource utilization, when the second uncertainty is looked upon, on the sudden completion of ready tasks, if the machines do not have any waiting tasks, they are disabled. The figure 4 shows this that on searching for other tasks to execute, if there are no waiting tasks, plus the next workflow tasks are'nt ready too, the service instance is turned off using the function optimizeVm(). Now in the next section we'll see through experiments how algorithm gives us considerable performance.

D. Experimental Setup:

The software used for the experiments is Eclipse Java Neon and WorkflowSim1.0. The configuration of the system used are 64-bit Windows 10 OS, Intel(R) Core(TM) i5 CPU 2.20 GHz, 8GB RAM. The proposed algorithm is developed in Java language. Then Workflow Sim simulator is employed to check the developed algorithm. It offers workflow level support during the simulation. WorkflowSim is that the toolkit used for simulation of scheduling algorithms. it's the advanced version of CloudSim by offering better workflow management and accurate evaluation. CloudSim [14] is employed for simulation of cloud services and infrastructure. CloudSim allows executing only single workload. It doesn't consider failures and overheads. It doesn't support clustering and job dependencies. Unlike other simulators, failures and overheads occurred within the heterogeneous system are considered into the WorkflowSim. It also supports clustering. Figure 5 shows the WorkflowSim Architecture. WorkflowSim contains multiple layers like Failure monitor, Failure generator, Clustering engine, Workflow engine and Workflow mapper along side the Workflow scheduler which is present into the CloudSim. Workflow mapper has used for mapping non-concrete workflows to the particular workflows which are reliant on execution sites. Data dependencies are managed by Workflow engine. Tasks are scheduled to the resources with the assistance of Workflow Scheduler. Small jobs are combined into an outsized one by using Clustering engine.

 HUGE WORKFLOWS COMPUTE plstij AND plfti FOR EACH TASK tij IN Wi TASK tij is ready? NO PLACE tij IN TASK POOL YES SORT READY TASKS BY plstij IN INCREASING ORDER mapReadYTask()	 Su HAS WAITING TASK YES EXECUTE FIRST WAITING TASK NO SEARCH tij 'S SUCCESSOR TASKS IN TASK POOL TASKS ARE READY? NO optimizeVM() YES mapReadyTask()
Figure 3. Algorithm process followed when new workflows arrive	**Fig. 4. Algorithm process followed when a task completes suddenly**

V. RESULT AND DISCUSSION

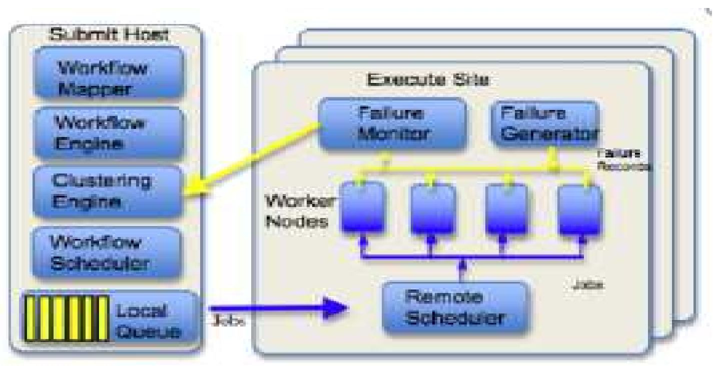


Fig. 5. Workflowsim Architecture

To test the proposed algorithm, following configuration of virtual machines are used(shown in below table).

Table-I: Virtual Machines configurations used in simulation.

Service type(u)	Type name	Price($)	F(u)
1	Small	0.023	8
2	Medium	0.464	4
3	Large	0.0928	2
4	xLarge	0.1856	1

For experimenting, these four scientific workflow applications are used: Montage (used in astronomy), Cyber Shake (used in earthquake science), SIPHT (used in bioinformatics), LIGO Inspiral (used in researching gravitational physics) [16]. For each workflow class, the structure of the workflow is shown in Table 2, which is taken from the Pegasus Workflow repository [17]. It gives the description of the four applications which will be used in experiment in the next chapter.

Table-III: Service renting costs for different workflow algorithms at different deadline base

Deadline base	I-TOF	TOF	FCFS
2	1600	1780	2500
4	1400	1787	2520
6	1453	1726	2550
8	1339.78	1710	2573
10	1100	1780.7	2571

E. Service renting cost evaluation

The first experimental result's administered by comparing the proposed i- TOF with TOF [1] and a standard algorithm FCFS fluctuating the parameter deadline base to think about the service renting cost constraint. Table 4 shows the service renting costs that are read at different deadline base for the three algorithms. it's seen in figure 7 that the service renting cost of all the algorithms decrease slowly with increasing deadline base. the rationale being that, as we increase the deadline, the schedulers search for the cheaper machines since it's got longer time to execute an equivalent task. it's clearly visible that the i-TOF algorithm outperforms the opposite two algorithms and provides the reduced cost. While resource overprovisioning can cost users quite necessary, resource under provisioning hurts the appliance performance. the value effectiveness of cloud computing highly depends on how well the customer can optimize the value of renting resources (VMs) from cloud providers. the difficulty of resource provisioning optimization from cloud-consumer potential may be a complicated optimization issue, which incorporates much uncertainty parameters. there's a way research avenue available for solving this problem because it is within the real-world. Here, during this paper we offer details about various optimization techniques for resource provisioning. Reserved instances are cheapest resources. Price is predicated on the amount of subscription (static). Customer must reserve the resources beforehand. Customer might overpay for the resources reserved if he/she doesn’t use them extensively and he might under buy the resources reserved for while [6]. On-demand instances are the very best priced resources. Price is about by the service provider and remains constant. Consumer has got to pay per use. Customer is conscious of the precise price to be paid. Resources are reserved for the customer for the paid period of your time [7]. Service provider might reserve the resources for extended than the customer’s utilized. Service provider cannot raise the worth when demand is high; when demand is low, the user pays above the market value. Spot Instances [8] allow you to specify the utmost hourly price that you simply are willing to pay to run a specific instance type, usually less than the On-Demand rate. These are suitable for both customers and therefore the service provider because the worth is about consistent with the extent of supply and demand. Less scalability of high demand within the market than fixed pricing, the spot instances are the unused on-demand instances. The cash price fluctuates supported supply and demand for instances, but customers will never pay quite the utmost price they need specified. If the cash price moves above a customer’s maximum price, the customer’s instance are going to be pack up by the cloud provider [9]. Figure 1 shows three pricing models.

The second experimental result is carried out by comparing the proposed i-TOF, TOF and algorithm FCFS to optimize the considered QOS constraint- service renting cost constraint. The i-TOF algorithm outperforms than TOF as it gives the reduced cost with different types of scientific applications having same number of tasks(100). Table 4 shows what the service renting costs come for different workflow applications using all three scheduling algorithms. Figure 8 shows the clear comparison. It is clearly seen that the i- TOF outperforms all the other algorithm and gives least service renting costs and the FCFS gives highest cost

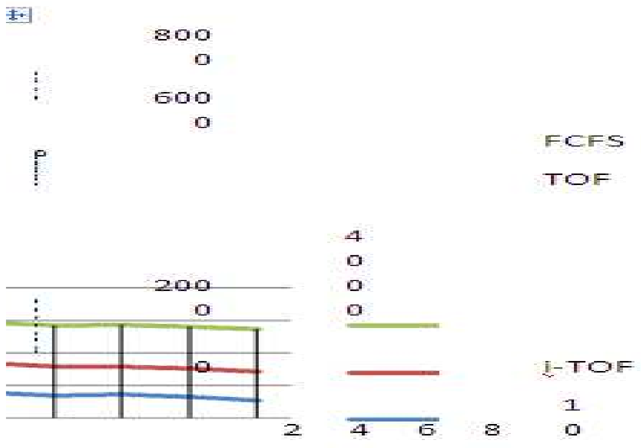


Fig. 7. Comparison of algorithms by single workflow application with increasing deadline base.

Table-IV: Service renting costs for different workflow algorithms at different deadline base

Deadline Base	i-TOF	TOF	FCFS
2	1600	1780	2500
4	1400	1787	2520
6	1453	1726	2550
8	1339.78	1710	2573
10	1100	1780.7	2571

F. Resource Utilization evaluation

The third experimental result is carried out by comparing the proposed i-TOF with TOF[1] and a traditional algorithm FCFS fluctuating the parameter deadline base to consider the second constraint of resource utilization constraint. It is seen that the resource utilization of all the algorithms individually increase slowly with increasing deadline base. The reason being that, when the deadline is early, the scheduler maps the task to a machine having high processing speed and when the deadline is more, the slower machines too get the tasks. Table 5 gives the resource utilization readings in simulation cloud using different scheduling algorithms on deploying a single application. Looking comparatively, it is clearly visible in figure 9 that the i-TOF algorithm gives higher resour5e utilization than the other two algorithms.

Table-V: Service renting costs for different workflow algorithms at different deadline base.

Deadline base	i-TOF	TOF	Fcfs
2	0.6291	0.5231	0.34
4	0.698	0.5322	0.378
6	0.7231	0.6487	0.432
8	0.778	0.6772	0.491
10	0.891	0.7623	0.541

VI. CONCLUSION AND FUTURE WORK

This study strives to propose a better uncertainty aware TOF Work Flow Scheduling algorithm concerning the service renting cost and resource utilization by considering the uncertainties in cloud service environment. Experimental results convey that the approached architecture for cloud service platforms outperforms all the other algorithms. In the context of real-workflow traces, three experiments were carried out to prove the superiority of the proposed algorithm. To be precise, the i -TOF performs approximately 50% and 30% better in terms of service renting cost and resource utilization respectively. Presently, cloud service is chanced to have problem of resource failure [18], [19]. The expensive commodities maintained by cloud providers need to be protected for their effectiveness in case a failure occurs. Hence, the research can be further carried out considering the fault tolerance and robustness in scheduling workflows considering these uncertainties.

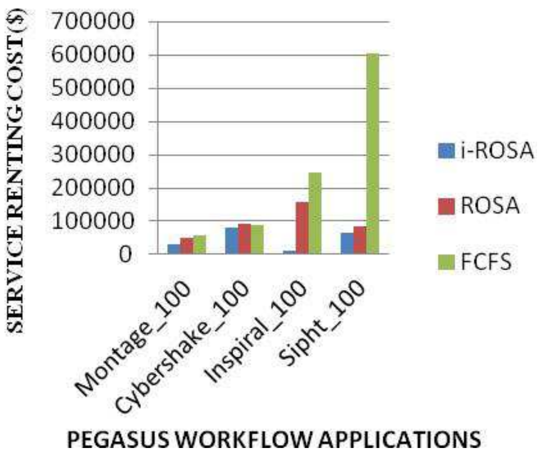


Fig. 8. Comparison of algorithms by different pegasus workflow applications with same number of tasks

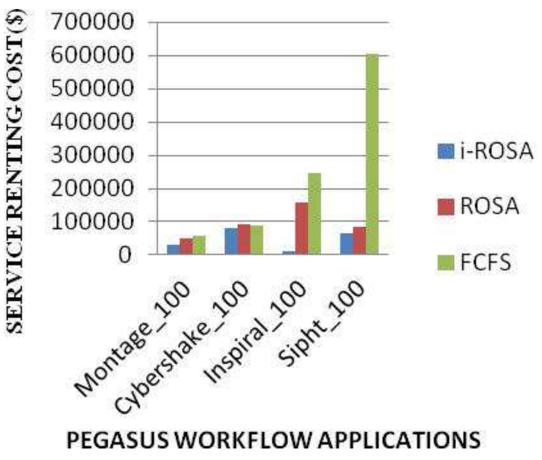


Fig. 9. Comparison of algorithms by single workflow application with increasing deadline base

REFERENCE

[1] H. Chen, X. Zhu, G. Liu and W. Pedrycz, "Uncertainty-Aware Online Scheduling for Real-Time Workflows in Cloud Service Environment," in IEEE Transactions on Services Computing.

[2] Gideon Juve, Ann Chervenak, EwaDeelman, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. 2013. Characterizing and profiling scientific workflows. Future Gener. Comput. Syst. 29, 3 (March 2013), 682-692. DOI= http://dx.doi.org/10.1016/j.future.2012.08.015

[3] F. Fakhfakh, H. H. Kacem and A. H. Kacem, "TOF Work Flow Schedulingin Cloud Computing: A Survey," 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, Ulm, 2014, pp. 372-378.

[4] Dharma P. AgrawalW. Zhao and J. A. Stankovic, "Performance analysis of FCFS and improved FCFS scheduling algorithms for dynamic real-time computer systems," [1989] Proceedings. Real-Time Systems Symposium, Santa Monica, CA, USA, 1989, pp. 156-165.

[5] Anousha S., Ahmadi M. (2013) An Improved Min-Min Task Scheduling Algorithm in Grid Computing. In: Park J.J..H., Arabnia H.R., Kim C., Shi W., Gil JM. (eds) Grid and Pervasive Computing. GPC 2013. Lecture Notes in Computer Science, vol 7861. Springer, Berlin, Heidelberg

[6] KobraEtminani, "A Min-Min Max-Min Selective Algorithm for Grid Task Scheduling," 3rd IEEE/IFIP International Conference in Central Asia on Internet 2007 (ICI 2007), Iran, September 2007.

[7] "Efficient TOF Work Flow SchedulingAlgorithm for Cloud Computing System: A Dynamic Priority-Based Approach " Gupta, I., Kumar, M.S. & Jana, P.K. Arab J Sci Eng(2018) 43: 7945. https://doi.org/10.1007/s13369-018-3261-8

[8] C. Zhou and S. K. Garg, "Performance Analysis of Scheduling Algorithms for Dynamic Workflow Applications," 2015 IEEE International Congress on Big Data, New York, NY, 2015, pp. 222-229.doi: 10.1109/BigDataCongress.2015.39

[9] Z. Cai, X. Li and J. N. D. Gupta, "Heuristics for Provisioning Services to Workflows in XaaS Clouds," in IEEE Transactions on Services Computing, vol. 9, no. 2, pp. 250-263, 1 March-April 2016.

[10] R. N. Calheiros and R. Buyya, "Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 7, pp. 1787-1796, July 2014.

[11] M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds," in IEEE Transactions on Cloud Computing, vol. 2, no. 2, pp. 222-235, 1 April-June 2014.

[12] Scheduling Precedence Constrained Stochastic Tasks on Heterogeneous Cluster Systems IEEE TRANSACTIONS ON COMPUTERS, vol. 63, no. xx, 2014

[13] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang, and L. Liu, "Fault tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 12, pp. 3501–3517, 2016.

[14] A. and Buyya, R. (2011). Cloudsim: a tool kit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience 41(1):23–50.

[15] Chen, W. and Deelman, E. (2012). Workflow sim: A toolkit for simulating scientific work- flows in distributed environments, E-Science (e-Science), 2012 IEEE 8thInternational Conference on, IEEE, pp.1–8.

[16] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," Future Generation Computer Systems, vol. 29, no. 3, pp. 682–692, 2013.

[17] R. N. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 7, pp. 1787– 1796, 2014.

[18] Z. Wen, J. Cala, P. Watson, and A. Romanovsky, "Cost effective, reliable and secure workflow deployment over federated clouds,"IEEE Transactions on Service Computing, vol. 10, no. 6, pp. 929–941, 20