EVOLUTIONARY MULTI-GOAL WORKFLOW PROGRESS IN SHADE

Mrs.M.Sangeetha [1] M.Tech, R.Nivi Priya[2], M.Ramya[3],V.Sowndarya[4]

[1](Assistant professor,Department of Computer Science and Engineering, Panimalar Engineering College,India)

[2] (Department of Computer Science and Engineering, Panimalar Engineering College,India)

[3] (Department of Computer Science and Engineering, Panimalar Engineering College,India)

[4] (Department of Computer Science and Engineering, Panimalar Engineering College,India)

---***---

Abstract— *Cloud computing provides promising platforms for executing massive programs with significant computational assets to offer on demand. Even though there are many current workflow scheduling algorithms in traditional allotted or heterogeneous computing environments, they have got difficulties in being without delay implemented to the Cloud environments. Considering that Cloud differs from conventional heterogeneous environments through its service-based totally aid managing method and pay-in line with-use pricing strategies in this paper, we highlight such problems, and version the workflow scheduling problem which optimizes each make span and price as a Multi-objective Optimization problem (MOP) for the Cloud environments. We recommend an evolutionary multi-objective optimization (EMO)-primarily based set of rules to resolve this workflow scheduling hassle on an infrastructure as a service (IaaS) platform. Novel schemes for problem-particular encoding and population initialization, health evaluation and genetic operators are proposed on this algorithm. The results additionally show that our algorithm can attain significantly higher solutions than existing modern day QoS optimization scheduling algorithms in most instances.*

Index terms: *Cloud computing, infrastructure as a service, multi-objective optimization, evolutionary algorithm, wor kflow scheduling.*

I. Introduction

In trendy years, Cloud computing has end up well-known and reached adulthood able to presenting the promising structures for website hosting massive-scale applications. In a Cloud model, on-name for computational resources, e.g., networks, storage and servers, may be allocated from a shared resource pool with minimum manage or interplay. The authors of this definition describe 3 service fashions in cloud computing: infrastructure as a provider (iaas), that encompass it offerings as e. g. computing electricity and storage ability; platform as a carrier (paas) that provide developer structures and software as a carrier (saas), which encompass software program services which are accessed via annet browser[1] [4].

With the help of these three services we use DAG [4]concept, an software model for describing workflow scheduling of workflows in grid allows mapping of responsibilities on heterogeneous assets according to a fixed of procedural regulations. Dynamism of resources in grid is an important trouble at the same time as making scheduling decisions, in which resources can fail necessarily. Screw ups of assets have damaging effects on overall performance of workflow application. Scheduling is the NP-tough problem; so many heuristic approaches had been implemented in the grid workflow [4]. One of the primary motives of any grid gadget is to meet consumer requirements in an intuitive manner by means of thinking about a couple of goals or criterion. Many specific criterion can be taken into consideration in scheduling of complicated workflow [6]computational tasks, generally encompass execution time of the assignment, value of the venture to run on a resource, utilization of resources, reliability, turnaround time and plenty of others. ho, et al [4] proposed the ordinal optimization (oo) method for discrete-occasion problems with very large solution space. Sooner or later, they [5] demonstrated that the oo method is powerful to generate a smooth or suboptimal technique to most np-difficult issues.

II. EXISTING SYSTEM

When scheduling workflows, the characteristics that make cloud range from grid or other conventional heterogeneous environments include 1) the complicated pricing schemes and a couple of the large-length useful resource swimming pools. A great deal existing paintings at the workflow scheduling hassle assumes that the financial price for a computation is based on the quantity of actually used resources. As an instance, posh assumes that the cost for executing a assignment is linearly or exponentially correlated to the overall variety of used cpu cycles. With this assumption, a few essential corollaries are 1) the whole fee of a workflow is the sum of the fees of all sub-responsibilities, and 2) the price of a assignment is constant whilst running on certain carrier and 3) it does no longer display start and destination time for processing a report and 4)it does now not specify how the facts are stored and manipulated. But, in cloud pricing schemes, the cost is decided by using the walking time of the underlying web hosting times. Also, the runtime is commonly measured with the aid of counting fixed-size time periods, with the partially used intervals rounded up. such schemes make the fee due to a project difficult to be precisely expected before scheduling. For example, a undertaking that stocks the equal time c programming language with the previous project hosted inside the identical instance might not produce greater cost. However, for a project which starts off evolved a brand new time c language however does now not use it entirely, the value might be extra than the anticipated.

III. PROPOSED SYSTEM

A cloud-aware extension to make list-based heuristics can be utilized in cloud is proposed. This extension constructs a constrained-length example pool with the capacity to host all viable schedules from cloud in advance. In order to agenda a 10-task workflow, a set containing 10 instances for each example kind is ready. An iaas platform offers computational resources through the virtual machines. A running digital system is known as an instance. It's miles commonplace for a iaas platform to provide a extensive variety of instance sorts comprising various combos of cpu, memory and network bandwidth. in this paper, cpu capacities, which determine the real execution time of obligations, and bandwidths, which affect the data transformation time, are taken into consideration for each example type.

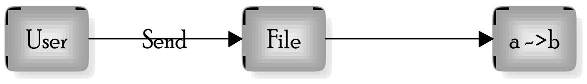
IV. SYSTEM MODULES:

USER MODULES:

It is the first interface that appears on the screen when the application is being loaded. This interface displays the name of the application and some other information about the software. The page consists of logins that exist for several other levels in the application. They consist of administrator, scheduling and algorithms.

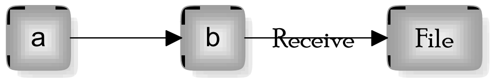
SCHEDULING TASKS:

First, we highlight the demanding situations for present scheduling algorithms to be without delay implemented to cloud, and formulate the cloud workflow scheduling problem with actual-global cloud traits. Those challenges rise up from the differences between cloud and the traditional heterogeneous environments together with grid, and the truth that maximum of the existing algorithms nevertheless expect that the heterogeneous environments are grid-like. Moreover, we design our set of rules with the goal of being able to be directly used within the iaas environments. To the pleasant of our know-how, the proposed set of rules is the first multi-goal workflow scheduling algorithm which considers the actual-international pay consistent with- use pricing strategies and on the equal time has been designed at once primarily based on the example-based totally iaas version



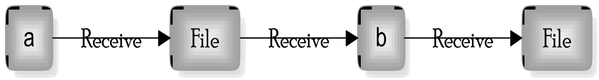
V. DAG ALGORITHM:

In which the tasks have been indexed using the results of a topological sort. Gives the encoding of a possible schedule for this workflow. In this schedule, the fitness function, discussed in follows the sequence ½T0; T1; T3; T5; T2; T4; T6_to compute the finish time of T6, which is used as the make span of the workflow. It gives the mappings from the tasks to the instances and from the instances to their types of task T0 will be scheduled to instance I1whose type is P4.C



VI. WORKFLOW SCHEDULING:

In the workflow scheduling problem, the fitness of a solution is related to a trade-off between two objectives which are make span and cost. As calculating the make span of a solution is to compute the finish time of Taxi. Here we define two functions ST and FT, which are respectively the start time and finish time of Ti in a given schedule. The start time of a task depends on the finish time of all its predecessors ,the communication time between its predecessors and itself, and the finish time of the previous task that has been executed on the same instance.



VII. WORKFLOW SCHEDULING PROBLEM WORKFLOW DEFINITION:

A common method to represent workflow is to use Direct Acyclic Graph(DAG). A workflow is a DAG W=(T,D), where T={ T0,T1….Tn} is the set of tasks and D={(TI,TJ)|TI,TJ belongs to T.} is the set of data or control dependentcies, The assigned to the tasks represent their reference execution time, which is the time to running the task on a processor of a specific type, and the weights attached to the edges represent the size of the data transferred between tasks. The reference exection time to time Ti is denoted as refertime (ti)and the data transfer size from ti to tj is denoted as data as (Ti,Tj).

In addition, we define all predecessors of tasks Ti, as

Pred(Ti)={Tj|(Tj,Ti)belongs to D}

For a given W ,T entry denotes an entry tasks satisfying

Pred(Tentry)=null

And T exit denotes an exit tasks satisfying

Not Ti belongs T: exits which belongs to pred(Ti).

Most scheduling algorithms require a DAG with a single T entry and single T exist. This can be easily assured by adding a pseudo T entry and T exit with zero weight to the DAG. In this paper, we also assume that the given workflow has single T entry and T exit.

WORKFLOW SCHEDULING PROBLEM:

Given a workflow W=(T,D) and an Iaas platform S=(I,P,M) a scheduling problem is to produce one or more solutions R=(Ins, Type, Order)where Ins and Type are mappings indicating which instance each task is put on the type of that instance, as

Ins:TI,$\rightarrow$I,Ins(Ti)=Ij;

Type:I, $\rightarrow$ P,Type(Is)=Pt;

And order is a vector containing the scheduling order of tasks. An order must satisfy the dependency restrictions between tasks, that is, a task cannot be scheduled unless all its predecessors have been scheduled. In this paper, we consider the problem that uses only one pricing option in a single schedule. The pricing option is chosen by users, denoted as Mo.Combining several pricing in our future work.

EVOLUTIONARY MULTI-GOAL OPTIMIZATION:

A multi-objective optimization problem is a problem that has several conflicting objectives which need to be optimized simultaneously :

Minimize:F(x)=(f1(x),….f2(x),fk(x))^T

Where x belongs X and X is the decision space. The workflow scheduling problem can be seen as an MOP, whose objectives in an MOP usually conflict with each other, ***parent dominance*** is commonly used to compare solutions. For u, v belongs to X, u is said to *dominate* **v** if and only if,

Fi(u)<=fi(u)^belongs j:for all j:fj(u)<fj(v).

A solution x* is *pareto optimal* if it is not dominated by any other solution .The set of all pareto optimal solutions in the objective space is called *pareto front*. For the Cloud workflow scheduling problem, schedule I* dominates schedule I if neither the cost nor the makespan of I* is larger than that of I, and at least one them is less. EAs which simulate natural evolution processes have been found increasing successful for addressing MOPs with various characteristics [12],[13],[14],[15].One significant advantage of EAs in the context of MOPs(called EMO Algorithm) is that they can achieve an approximation of the Pareto front ,in which each solution represents a unique trade off amongst the objectives.

Due to the properties of the cloud workflow scheduling problem, it is hard to adopt the existing genetic operations in the EMO areas, such as binary encoding ,real-valued encoding and the corresponding variations operators based on them. By taking full advantage of the problem's properties, we thus present a whole set of the exploration operations, including encoding, population initialization, crossover, and mutation. These operations can work with any explitation operations in the EMO area, as we have already applied them to several classical EMO area , as we have already applied them to several classical EMO algorithm such as NSGA-2,SPEA2,and MOEA/D .Some algorithms used for scheduling are listed below:

1.fitness function

2.Encoding

3.Genertic operators

3.1 cross over

3.2 mutation

4.Initial population

1.FITNESS FUNCTION:

In the workflow scheduling problem, the fitness of a solution is related to a trade-ff between two objectives which are makespace and cost. Here we define two functions ST and FT, which are respectively the *start time* and *finish time* of Ti in a given schedule. The start time of a task depends on the finish time of all its predecessors, the communication time between its predecessors and itself, and the finish time of the previous task that has been executed on the same instance. The recurrence relations are ,

ST(Tentry)=0, (1)

ST(Tj)=max{avail(Ins(Ti)),max
(FT(Tj)+Timecomm(Tj,Ti))}, (2)

FT(Ti)=ST(Ti)+Timecomp(Ti) (3)

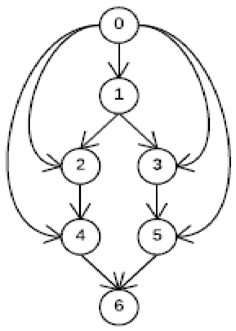


Figure 1 an example of workflow DAG

Where ,avail(Ii) is the available time of instance Ii, which changes dynamically during scheduling. After Ti is decided to be scheduled to the instance (Ij)avail(Ij) will be updated to FT(Ti).

After the finish time of T_{exit} is calculated, the final available time of an instance will be used as the estimate of its shutdown time, and the start time of the first task being assigned to the instance will be used as the estimate its launch time. The separate costs of all the instance being used are then calculated by the platform-specific charge function and summed up as the total cost.

2.ENCODING:

Here, the first step of encoding is to make a topological sorting then assign an integer index to each task according to the sorting results. The index starts from 0, and Ti is referred to a task whose index is i. As discussed in Section 3.3, a solution is a three-tuple containing a sequence Order and two mappings Ins and type. We split a chromosome into three strings to represent them respectively. The string order is a vector containing a permutation of all task indexes. If i occurs before j in order, the hosting instance of task Ti will be determined before that of Tj. However, it does not mean that the execution of Ti must start before Tj. The start time of a task is determined by the hosting instance and its predecessors (see Eq. (2)). The second string task2ins is a n-length vector representing the mapping Ins, in which an index represents a task and its value represents the instance where this task will be executed. As mentioned in Section 3.2, the instance set I could be reduced to a n-size set, so that it is possible to index all instances using integers from 0 to n _ 1. For example, task2ins[i]=j makes Ti be assigned.

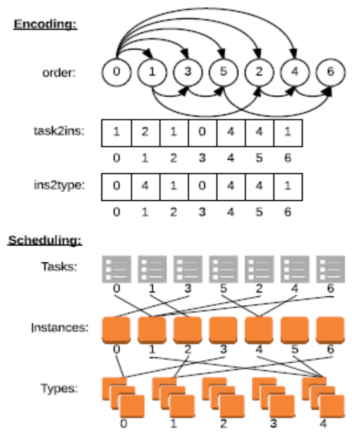


Fig. 2. Encoding and scheduling scheme of a valid schedule for the DAG

to the instance with index j (represented as Ij).The instance types are also indexed previously using integers from 0 to m _ 1, and ins2type[j]=k indicates that the type of instance Ij is Pk. Fig. The 1 shows an example DAG, in which the tasks have been indexed using the results of a topological sort. Fig. 2 gives the encoding of a possible schedule for this workflow. In Section 4.1, follows the sequence ½T0; T1; T3; T5; T2; T4; T6_

to compute the finish time of T6, which is used as the make span of the workflow.

VIII. GENETIC OPERATORS:

Cross over

A valid scheduling order must follow the task dependencies.For example, if a task T_ is a successor of T, T_ must occur after T in string order. The crossover operation should not violate these restrictions. We design the crossover operator for the order strings as given in Fig. 3. First, the operator randomly chooses a cut-off position, which splits each parent string into two substrings.(Step 3). After that, the two first substrings are swapped to be the offspring, and the second substrings are discarded.(Steps 4-5). Then, each parent order string is scanned from the beginning, with any task that has not occurred in the first substring being appended to the end of this offspring (Steps 6-10, 11-15). This operator will not cause any dependency conflict since the order of any two tasks should have already existed in at least one parent. An example of this operation is given in Fig. 4, in which the position 3 is randomly chosen as the cut-off position. The first three items in both strings are swapped. Then, the missing tasks for each offspring are appended to its end, in their original orders. Analogously, the operator first randomly selects a cut-off point, and then, the first parts of two parent task2ins strings are swapped. Here, it is noteworthy that the type of the instance on which a task is running could also be important information for this task, and it is better to keep this relationship. Mutation will be introduced to increase the search ability of the algorithm.

```
1:  procedure CROSSOVERORDER($A, B$)
2:      $n \leftarrow$ number of tasks
3:      $p \leftarrow$ RandInt$(0, n-1)$
4:      $order_a \leftarrow$ SubString$(B, 0, p)$
5:      $order_b \leftarrow$ SubString$(A, 0, p)$
6:      for all $T$ in $A$.order do
7:          if $T$ not in $order_a$ then
8:              append $T$ to the end of $order_a$
9:          end if
10:     end for
11:     for all $T$ in $B$.order do
12:         if $T$ not in $order_b$ then
13:             append $T$ to the end of $order_b$
14:         end if
15:     end for
16: end procedure
```

Fig. 3. Crossover operator for string order. This operator in place modifies the order strings of individuals A and B to produce two offspring.

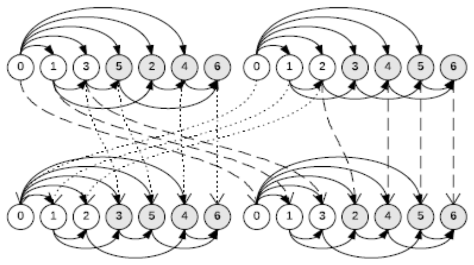


Fig. 4. An example of order crossover.

The pseudocode of this operation is given in Fig. 5. Step 3 selects the cut-off point. Before swapping tasks in the first parts (Step 7), an ancillary procedure is invoked. This ancillary procedure, called DecideType, decides on the type of the new hosting instance of task T in individual B, when moving from the instance specified in individual A. For this decision, the types of the new instance (I0), in both individuals, are taken out (Pa and Pb in Steps 2-3). Then Step 3decides whether the type of I0 in B should be changed to Pa or not. If there is no any task whose index is greater than or equal to the cut-off position p is scheduled to I0 (Step 4), the type of I0 will be changed to Pa (Step 10), with a mutation.

```
1:  procedure CROSSOVERINS($A, B$)
2:      $n \leftarrow$ number of tasks
3:      $p \leftarrow$ RandInt$(0, n-1)$
4:      for $i \leftarrow 0, \ldots, p-1$ do
5:          DecideType$(T_i, A, B, p)$
6:          DecideType$(T_i, B, A, p)$
7:          Swap$(A.$task2ins$[T_i], B.$task2ins$[T_i]$
8:      end for
9:  end procedure
1:  procedure DECIDETYPE($T_i, A, B, p$)
2:      $I' \leftarrow A.$task2ins$[T_i]$
3:      $P_a, P_b \leftarrow A.$ins2type$[I'], B.$ins2type$[I']$
4:      if $\exists j : j \geq p \wedge B.$task2ins$[T_j] = I'$ then
5:          if $P_a \neq P_b$ then
6:              $P \leftarrow$ RandChoice$(\{P_a, P_b\})$
7:              $B.$ins2type$[I'] \leftarrow P$
8:          end if
9:      else
10:         $B.$ins2type$[I'] \leftarrow P_a$
11:         mutate $P_a$ with a small probabiity
12:     end if
13: end procedure
```

Fig. 5. Crossover operator for strings Task2Ins and Ins2type and ancillary procedure DecideType. The CrossoverIns procedure in place modifies task2ins strings of individual A and B to produce offspring.

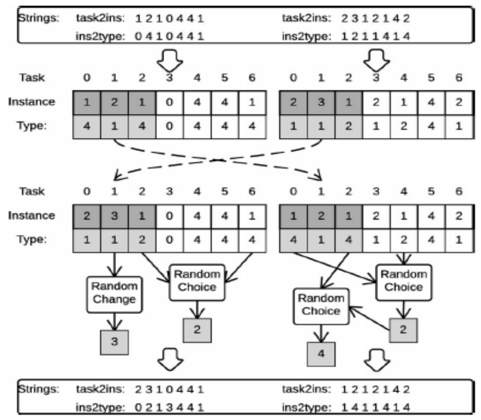


Fig. 6. An example of task2ins and ins2type crossover.

MUTATION:

Like the crossover operators, the mutation operator of string order should not break the task dependencies either. First, we define all successors of task Ti as

Succ(Ti)= {Tj | (Ti; Tj)belongs to D}

Fig. 7 gives the pseudocode of order mutation. Starting from task T, the operator searches for a substring in which each task is neither a predecessor nor a successor of T (Steps 4-10). Then, T is moved to a randomly chosen new position inside this substring (Steps 11-12). On each direction, the search procedure starts from the position of T, and stops once the current task is either in predðTÞ or in succðTÞ. Fig. 8 demonstrates an example where task 2 is randomly chosen to be the mutation point. A search is then performed to find the substring meeting the conditions, between task 1 and task 4. Finally, task 2 is randomly moved to a new position inside this substring.

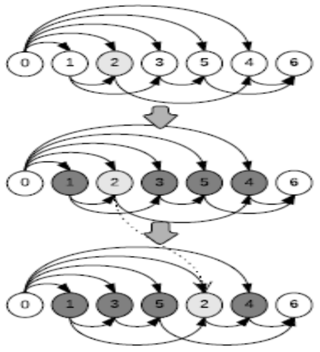


Fig 8.An example of order mutation.

Here, the mutation for the strings task2ins andins2type is performed by a classical operator, that is, randomlygenerating a new valid value for each position, witha small probability.

```
1: procedure MUTATEORDER($X, pos$)
2:     $n \leftarrow$ number of tasks
3:     $T \leftarrow X.\mathtt{order}[pos]$
4:     $start, end \leftarrow pos$
5:     while $start \geq 0 \wedge X.\mathtt{order}[start] \notin \mathrm{pred}(T)$ do
6:         $start \leftarrow start - 1$
7:     end while
8:     while $end < n \wedge X.\mathtt{order}[end] \notin \mathrm{succ}(T)$ do
9:         $end \leftarrow end + 1$
10:    end while
11:    $pos' \leftarrow$ RandInt$(start + 1, end - 1)$
12:    Move $T$ to $pos'$ in $X$.order
13: end procedure
```

Fig. 7. Mutation operator for order strings. Given a position pos, this operator randomly moves the pos^{th} task in $X.order$ to another valid position.

IX. INITIAL POPULATION:

In the workflow scheduling problem, the search space of solutions is typically huge, especially when a large workflow is involved, which could cause evolutionary algorithms very slow to coverage. In our algorithm, to accelerate the search procedure, the initial population consists of the individuals generated by different initialization methods. Assuming the size of population is n, these individuals include a schedule computed by HEFT, which is treated as fastest schedule, a ”cheapest” schedule produced at the same time, when executing HEFT, as an estimate of the cheapest schedule,n-2 random schedules initialized by a procedure named R and Type or Ins. First ,HEFT is slightly extended for guessing an individual that can approach the cheapest cost, along with the standard procedure of finding the fastest schedule. This cheapest schedule is produced by assigning the task to the instance which can minimize the currently-generated cost in the processor selection phase of each task. This individual might not be the actual cheapest one: inspite of that ,it could still be seen as a rough approximation of one endpoint of the Pareto front. At the same time, the fastest individual produced by the original HEFT could be used as another approximate endpoint. Besides these two heuristic-generated schedules, we initialize other individuals randomly. For each individual, the procedure is presented in fig.9. First, the string order is simply constructed as an increasing sequence ½0; 1; . . . n _ 1_ (line 4). Then , a specific instance type is randomly chosen, and all instances will share this type, by setting all bits of the ins2type string to the index of this type (line 5). Finally , the string task2ins is initialized by a random choice of two methods, with equal probability

(line 6). The first method is to put all tasks in a single instance, by setting all bits of task2ins to 0.

```
1:  procedure RANDTYPEORINS
2:      $n \leftarrow$ number of tasks
3:      $m \leftarrow$ number of instance types
4:      order $\leftarrow [0, 1, ..., n-1]$
5:      ins2type $\leftarrow$ replicate($m$, RandInt($0, m-1$))
6:      if Rand(0,1) $< 0.5$ then
7:          task2ins $\leftarrow$ replicate($n$, 0)
8:      else
9:          for all $i \in [0, n-1]$ do
10:             task2ins[$i$] $\leftarrow$ RandInt($0, n-1$)
11:         end for
12:     end if
13:     sched $\leftarrow$ {order, task2ins, ins2type}
14: end procedure
```

Fig. 9. The RandTypeOrIns procedure.

X. COMPLEXITY ANALYSIS:

The time complexity for both CrossoverOrder and MutateOrder is O(n), where n is the number of tasks. The time complexity of the procedure CrossoverIns is O(n^2),because for each swapped instance in the string task2ins,an O(n) scan is needed to find whether it also hosts another task according to the opposite individual. The evaluation procedure for each individual has an O(e)time complexity. For a given DAG, the number of edges could be at most n2,so the time complexity of each evaluation is on the order of O(n2). Thus, the overall complexity of the evolution is on the order of O(kgn2), with k individuals in population and *g* generations. Besides the evolution procedure, when initializing the first population, HEFT is performed once. The HEFT algorithm has Oðsn2Þ complexity where s is the number of available services [17]. By using the Cloud-aware extension proposed in [16], a heterogeneous environment can be constructed by m*n instances in Cloud, where m is the number of instance types. Thus, HEFT has the time complexity of O(mn3) in our initialization scheme. Above all, the overall computational complexity of our proposed algorithm is on the order of O(mn3 + kgmn2).However, we would like to point out that, when executing HEFT in the population initialization procedure, a large number of redundant calculations could be eliminated or optimized if using proper data structures. Most instances in the simulated service pool are not used at all, and several unused instances are actually identical if they also share a same type. Also, in practice, m *n is usually much less then k * g. Therefore, the most time-consuming parts would still be the evolution procedures, with the complexity of O(kgn^2).

XI. RELATED WORK:

There had been some of efforts within the Grid community to broaden general-motive workflow management solutions. [2] Web Flow is a multileveled machine for high performance allotted computing[2].The Directed Acyclic Graph (DAG) [4] primarily based assignment graphs in parallel computing are said already in literature for scheduling trouble. QoS[6] aware heuristic has been proposed in for grid unbiased challenge scheduling. In Heterogeneous Earliest finish Time(HEFT) and Genetic Algorithms have been applied with extension for the ASKALON surroundings to solve medical workflow applications in grid. E.Tsiakkouri et al. suggested scheduling algorithms LOSS and advantage. LOSS makes adjustment inside the time table generated by means of a time optimized heuristics while advantage in a value optimized heuristic's agenda in the users' certain finances constraint[3]. In the paper, effectiveness of Evolutionary Algorithms over simulated annealing and Particle Swarm Optimization has been provided for scheduling jobs on computational Grids. Furthermore ,the Multi-objective Evolutionary Algorithms(MOEAs) for workflow scheduling were investigated to optimize conflicting goals concurrently to generate Pareto optimize solutions. Even as Strebel and degree as well as Kondo et al. attention on the provider perspective[3]. They developed a software tool to calculate placing-up and upkeep charges for a cloud(costs of hardware,software,strength ,cooling staff and real-property)[5]. In pl ace of focusing on physical hardware [1] they focus on maximum digital machines that can be deployed within a data center to react greater flexible on purchaser needs.

XII. CONCLUSION AND FUTURE WORK:

Although there are many existing workflow scheduling algorithms for the mulit-processor architectures or heterogenous computing environments, they have difficulties in being directly applied to the cloud environment. In this paper, we try to address this by modeling the workflow scheduling problem in cloud as a multi-objective optimization problem where we have considered the real-world cloud computing models. To solve the multi-objective cloud scheduling problem which minimizes both makespan and cost simultaneously, we propose a novel encoding scheme which represents all the scheduling orders, task-instance assignments and instance specification choices, based on this scheme,we also introduce a set of new genetic operators, the evaluation function and the population initialization scheme for this problem. We apply our designs to several popular EMO frameworks, and test the proposed algorithm on both the real workflows and work sets for randomly generated workflows. Combining several pricing options in a single scheduling procedure might be studied in our future work.

REFERENCE:

[1] Benedikt Martens, Marc Walterbusch and Frank Teuteberg,"Costing of Cloud Computing Services: A Total Cost of Ownership Approach"in Accounting and Information Systems.

[2] G. Bruce Berriman, John Good, Anastasia Laity,"Pegasus: a Framework for Mapping Complex ScientificWorkflows onto Distributed Systems".

[3] RituGarg, Awadhesh Kumar Singh," Multi-Objective Optimization to Workflow Grid Scheduling using Reference Point based Evolutionary Algorithm" International Journal of Computer Applications (0975 – 8887)Volume 22– No.6, May 2011

[4] A.K.M. KhaledAhsanTalukder, Michael Kirley and RajkumarBuyya,"Multiobjective Differential Evolution for WorkflowExecution on Grids".

[5]Fan Zhang, Junwei Cao, Kai Hwang and Cheng Wu," Ordinal Optimized Scheduling of Scientific Workflows in Elastic Compute Clouds"

[6] Maria Alejandra Rodriguez Sossa," Resource Provisioning and Scheduling Algorithms for Scientific Workflows in Cloud ComputingEnvironments".

[7] NavjotKaur, Taranjit Singh Aulakh, Rajbir Singh Cheema," Comparison of Workflow Scheduling Algorithms in Cloud Computing", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 10, 2011.

[8] OrachunUdomkasemsub, Li Xiaorong, TiraneeAchalakul," A Multiple-Objective Workflow Scheduling Framework for Cloud DataAnalytics".

[9] Bo CHENG," Hierarchical Cloud Service Workflow Scheduling Optimization," State Key Laboratory of Networking and Switching Technology, Beijing

University of Posts and TelecommunicationsSchema Using Heuristic Generic Algorithm.

[10] Thomas Feilhauer* and Martin Sobotka," DEF - a programming language agnostic framework and execution environment for theparallel execution of library routines",Feilhauer and SobotkaJournal of Cloud Computing: Advances, Systems and Applications (2016) 5:20

DOI 10.1186/s13677-016-0070-z

[11] Maria A. Rodriguez, RajkumarBuyya," Scientific Workow Management System for Clouds".

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, Apr. 2002.

[13] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," IEEE Trans. Evol. Comput., vol. 13, no. 2, pp. 284–302, Apr. 2009.

[14] H. Ishibuchi, Y. Hitotsuyanagi, N. Tsukamoto, and Y. Nojima, "Many-objective test problems to visually examine the behavior of multiobjective evolution in a decision space," in Proc. Parallel Problem Solving Nat., 2010, pp. 91–100.

[15] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for pareto-based algorithms in many-objective optimization," IEEE Trans. Evol. Comput., vol. 18, no. 3, pp. 348–365, Jun. 2014.

[16] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in amazon ec2," Cluster Comput., vol. 17, no. 2, pp. 169–189, 2014.