International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Target Based Resource Allocation and Scheduling Algorithm for Systematic Workflow in Cloud

Lokesha K. R.[1], Dr. K Raghuveer[2]

[1]MTech, CNE Prof & HOD
[2]NIE, Mysuru Information Science and Engineering, NIE, Mysuru

Abstract: ***Cloud computing is the latest emerging technology that provision service in distributed system supports to investigate its benefits on executing scientific scheduling workflow. However, the major goal of cloud computing is to allocate resource to user truly needed based on pay for use. Furthermore, deadline scheduling workflow is the most challenging problem in cloud. This paper proposes a system that includes minimize cost of the execution, resource provisioning, and deadline constrained scheduling scientific workflows on Infrastructure as a Service (IaaS) clouds. We use the concept of an algorithm based on the particle swarm optimization (PSO), deadline schedule generation. These algorithms minimize the overall workflow execution cost while meeting deadline constraints. The results show that our approach increase likelihood of deadlines being met and reduces the execution cost as compare to current state-of-the-art algorithms.***

Keywords: Cloud Computing, Resource Allocation, Scheduling Algorithm, IaaS

1. Introduction

Cloud computing [1] is the latest emerging tec0hnique in distributed computing environment that delivers hardware infrastructure, resource and software applications as services dynamically. The users can consume these services based on a Service Level Agreement (SLA) which defines their required Quality of Service parameters, on a pay for use basis over the internet. There are different types of cloud providers; each provider has different service product offerings. Cloud providers are decomposing into a hierarchy of as-a-service terms: Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). This paper mainly focuses on IaaS clouds which provision a virtual machine pool of heterogeneous, unlimited resources that can be accessed based on user demand. Although, they facilitate the flexibility of elastically acquiring or releasing resources with varying configurations to best suit the requirements of an application. Even though this enables the users and provides more control over the resources, and it also dictates the development of advance innovative scheduling techniques so that the distributed resources are efficiently utilized.

Workflows have been frequently used for describing a wide range of scientific applications in distributed systems [2]. Usually, a Graph represents the workflow in which each decompose task is represented by a node, and each control dependency between tasks is represented by an edge between the corresponding nodes. Due to the benefits of workflow applications, many Grid projects such as Pegasus [3], ASKALON [4], and GrADS [5] have designed a specific workflow management system to define, execute and manage the workflows on the Grid. There are two phases when planning the execution of a workflow in a cloud environment. The first phase is resource provisioning; in this phase, the resources computing that will be used to execute the tasks are selected and provisioned. In the second phase, a deadline schedule is computed using schedule generation algorithm and each task is mapped onto the best-suitable resource. The selection of the mapping and resources of the tasks is done so those different user defined qualities of service (QoS) requirements are met. Existing works in this field, especially those developed on Grids or Clusters, focused mainly on the scheduling phase. The reason for this is that these environments provide a static virtual pool of resources which are already available to execute the tasks and whose configuration is known in advance. Since this is not possible in cloud environments, both phases need to be combined and addressed in order to produce an efficient execution plan in workflow.

2. Related Work

A. Existing System

Workflow scheduling on distributed systems in a scientific method has been widely studied last few years and is NP-hard by a reduction from the multiprocessor problem in scheduling [7]. Therefore it is possible to find solution but impossible to generate an optimal solution within polynomial time and algorithms focus on generating approximate optimal solutions. There are number of algorithms have been developed that work towards to find a schedule that meets the user's QoS requirements.

The previous solutions provide a highly useful insight into the challenges and potential solutions for workflow scheduling for grid. Moreover they are not optimal solution for utility-like environments such as IaaS clouds. The analysis provides the various specific features of cloud environments that need to be considered when developing a scheduling algorithm. The different types of VMs are acknowledged that there are with different prices and that they can be leased on demand, depending on the application's requirements. Further, they tailor their approach so that the minimized the execution cost based on the cloud's pricing model.

B. Problem Definition

A static cost-minimization, deadline-constrained heuristic in a Cloud environment for scheduling a scientific workflow application. Our approach considers

fundamental features of IaaS providers such as the dynamic provisioning and heterogeneity of unlimited computing resources as well as VM performance variation.

To achieve higher efficiency, both resource provisioning and scheduling are merged and modeled as an optimization problem. PSO is then used to solve earlier mentioned problem and produce a schedule defining not only the In this our contribution on an algorithm with higher accuracy in terms of meeting deadlines at lower costs that considers heterogeneous resources that can be dynamically acquired and released and are charged on a pay-per use basis.

3. Proposed Work

A. Particle Swarm Optimization

There are two key steps when modeling a PSO problem. The first one is defining how the problem will be encoded, that is, defining how the solution will be represented. The second one is defining how the "goodness" of a particle will be measured, that is, defining the fitness function.

Particle's Position

Coord.1	Coord.2	Coord.3	Coord.4	Coord.5	Coord.6	Coord.7	Coord.8	Coord.9
1.2	1.0	2.1	3.0	1.2	1.1	2.0	2.2	2.3

Task to Resource Mapping

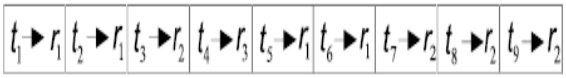

Figure 1: shows encoding of a particle's position.

ALGORITHM 1
PARTICLE SWARM OPTIMIZATION

1. Set the dimension of the particles to d
2. Initialize the population of particles with random positions and velocities
3. For each particle, calculate its fitness value
   - 3.1 Compare the particle's fitness value with the particle's $pbest$. If the current value is better than $pbest$ then set $pbest$ to the current value and location
   - 3.2 Compare the particle's fitness value with the global best $gbest$. If the particle's current value is better than $gbest$ then set $gbest$ to the current value and location
   - 3.3 Update the position and velocity of the particle according to equations 7 and 8
4. Repeat from step 3 until the stopping criterion is met.

The range in which the particle is allowed to move is determined in this case by the number of resources available to run the tasks. As a result, the value of a coordinate can range from 0 to the number of VMs in the initial resource pool. Based on this, the integer part of the value of each coordinate in a particle's position corresponds to a resource index and represents the compute resource assigned to the task defined by that particular coordinate. In this way, the particle's position encodes a mapping of task to resources. Following the example given in Fig. 1; there are three resources in the resource pool so each coordinate will have a value between 0 and 3. Coordinate 1 corresponds to task 1 and its value of 1.2 means that this task was assigned to resource 1. Coordinate 2 corresponds to task 2 and its value of 1.0 indicates that task 2 was assigned to resource 1. The same logic applies to the rest of the coordinates and their values.

B. Schedule Generation

$$ET_{t_i}^{VM_j} = I_{t_i}/(P_{VM_j}*(1 - deg_{vm_j})), \quad (1)$$

$$TT_{e_{ij}} = d_{t_i}^{out}/\beta, \quad (2)$$

$$PT_{t_i}^{VM_j} = ET_{t_i}^{VM_j} + \left(\sum_{1}^{k} TT_{e_{ij}} * s_k\right). \quad (3)$$

The pseudo-code to convert a particle's position into a schedule is shown in Algorithm 2. Initially, the set of resources to lease R and the set of task to resource mappings M are empty and the total execution cost TEC and time TET are set to zero. This is expressed as a matrix in which the rows represent the tasks, the columns represent the resources and the entry ExeTime (i, j) represent the time it takes to run task ti on resource rj. This time is calculated using Equation (1). The next step is the calculation of the data transfer time matrix. Such matrix is represented as a weighted adjacency matrix of the workflow DAG where the entry Transfer Time (i , j) contains the time it takes to transfer the output data of task ti to task tj. This value is calculated using Equation (2) and is zero whenever (i, j) or there is no directed edge connecting ti and tj.

The start time value ST(ti) is based on two scenarios. In the first case, the task has no parents and therefore it can start running as soon as the resource it was assigned to is available; this value corresponds to the current end of lease time of resource rpos[i], which is LETrpos[i]. In the second case, the task has one or more parents. In this situation, the task can start running as soon as the parent task that is scheduled to finish last completes its execution and the output data is transferred. However, if the resource is busy with another task at this time, the execution has to be delayed until such VM is free to execute ti.

The value of ET(ti) is calculated based on the total processing time and the start time of the task. To determine the processing time PT. we first need to compute the execution and the data transfer times. These two values are then added to obtain PT as defined in Equation (3). Finally we obtain the value of ET(ti) by subtracting ST(ti) from PT.

Now that we have computed all the elements of (ti, rpos(i), ST(ti), ET(ti)) we need to update two parameters associated to rpos(i) and add the resource to R if necessary. The first parameter is the time when the VM should be launched, LSTrpos(i) . However, if the resource is new and ti is the first task to be assigned to it then R is updated so that it contains the new resource rpos(i) and LSTrpos(i) is set to be equal to either the start time of the task or the VM boot time, whichever is bigger.

Once the algorithm finishes processing each coordinate in the position vector, R will contain all the resources that need to be leased as well as the times when they should be

started and shutdown. Additionally, the entire task to resource mapping tuples will be in M and each task will have a resource assigned to it as well as an estimated start and end times. With this information, the algorithm can now use Equations (4) and (5) to compute the execution cost TEC and time TET associated to the current solution. After this, the algorithm has computed R, M, TEC and TET and therefore it can construct and return the schedule associated to the given particle's position.

ALGORITHM 2
SCHEDULE GENERATION

Input: Set of workflow tasks T
Initial resource pool $\mathcal{R}_{initial}$
An array $pos[|T|]$ representing a particle's position
Output: A Schedule S

1. Initialize schedule components
   1.1. $\mathcal{R} = \emptyset, \mathcal{M} = \emptyset$
   1.2. $TEC = 0, TET = 0$
2. Calculate $ExeTime[|T| \times |\mathcal{R}_{initial}|]$
3. Calculate $TransferTime[|T| \times |T|]$
4. **for** $i = 0$ to $i = |T| - 1$
   4.1. $t_i = T[i], r_{pos[i]} = \mathcal{R}_{initial}[pos[i]]$
   4.2. if t_i has no parents
   $ST_{t_i} = LET_{r_{pos[i]}}$
   else
   $ST_{t_i} = max\left(max\left\{ET_{t_p} : t_p \in parents(t_i)\right\}, LET_{r_{pos[i]}}\right)$
   end if
   4.3. $exe = exeTime[i][pos[i]]$
   4.4. for each child t_c of t_i
   if t_c is mapped to a resource different to $r_{pos[i]}$
   $transfer += TransferTime[i][c]$
   end if
   end for each
   4.5. $PT_{t_i}^{r_{pos[i]}} = exe + transfer$
   4.6. $ET_{t_i} = PT_{t_i}^{r_{pos[i]}} - ST_{t_i}$
   4.7. $m_{t_i}^{r_{pos[i]}} = (t_i, r_{pos[i]}, ST_{t_i}, ET_{t_i})$
   4.8. $M = M \cup \{m_{t_i}^{r_{pos[i]}}\}$
   4.9. if $r_{pos[i]} \notin R$
   $LST_{r_{pos[i]}} = max(ST_{t_i}, bootTime)$
   $R = R \cup \{r_{pos[i]}\}$
   end if
   4.10. $LET_{r_{pos[i]}} = PT_{t_i}^{r_{pos[i]}} + LST_{r_{pos[i]}}$
5. Calculate TEC according to equation (4)
6. Calculate TET according to equation (5)
7. $S = (R, M, TEC, TET)$

$$TEC = \sum_{i=1}^{|R|} C_{VM_{r_i}} * \left\lfloor \frac{(LET_{r_i} - LST_{r_i})}{\tau} \right\rfloor, \quad (4)$$

$$TET = max\{ET_{t_i} : t_i \in T\}. \quad (5)$$

Finally, Algorithms 1 and 2 are combined to produce a near optimal schedule. In step 3 of Algorithm 1, instead of calculating the fitness value of the particle, we generate the schedule as outlined in Algorithm 2. Then we use TEC as a fitness value in steps 4 through 6 and introduce the constraint handling mechanism in step 4, ensuring that TET doesn't exceed the application's deadline.

4. Simulation Results

A. Deadline Constraint Evaluation

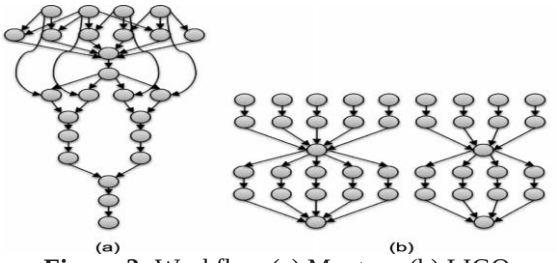

Figure 2: Workflow (a) Montage (b) LIGO

To analyze the algorithms in terms of meeting the user defined deadline, we plotted the percentage of deadlines met for each workflow and deadline interval. The results are displayed in Fig. 3. For the Montage workflow, ICPCP fails to meet all of the deadlines. PSO_HOM meets fewer than 50 percent of the deadlines on interval 1 but improves its performance on interval 2 and achieves a 100 percent hit rate for both intervals 3 and 4.

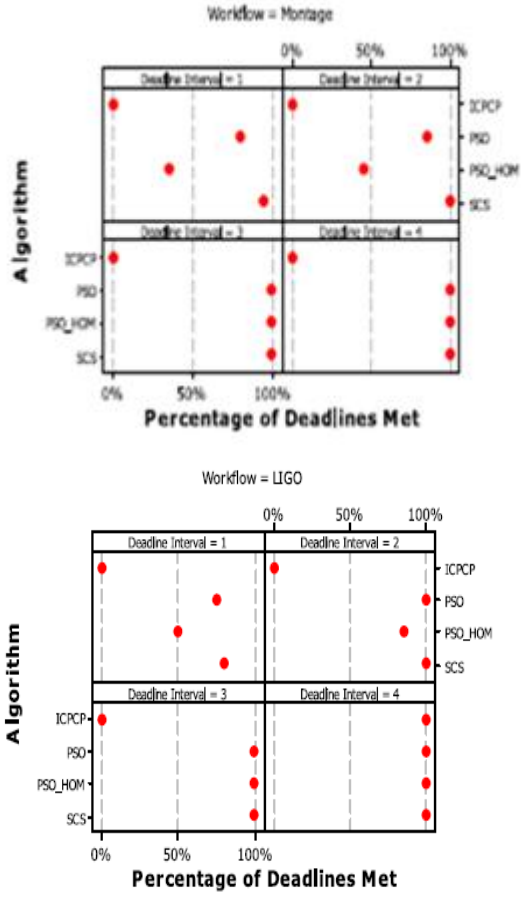

Figure 3: Individual value plot of deadlines met for each workflow and deadline interval

B. Cost Evaluation

The average execution costs obtained for each workflow are shown in Figure 4. We also show the mean make span as the algorithms should be able to generate a cost-efficient schedule but not at the expense of a long execution time. The reference line on each panel

displaying the mean is the deadline corresponding to the given deadline interval. We present this as there is no use in an algorithm generating very cheap schedules but not meeting the deadlines; the cost comparison is made Between PSO and SCS and SCS shows better result than PSO.

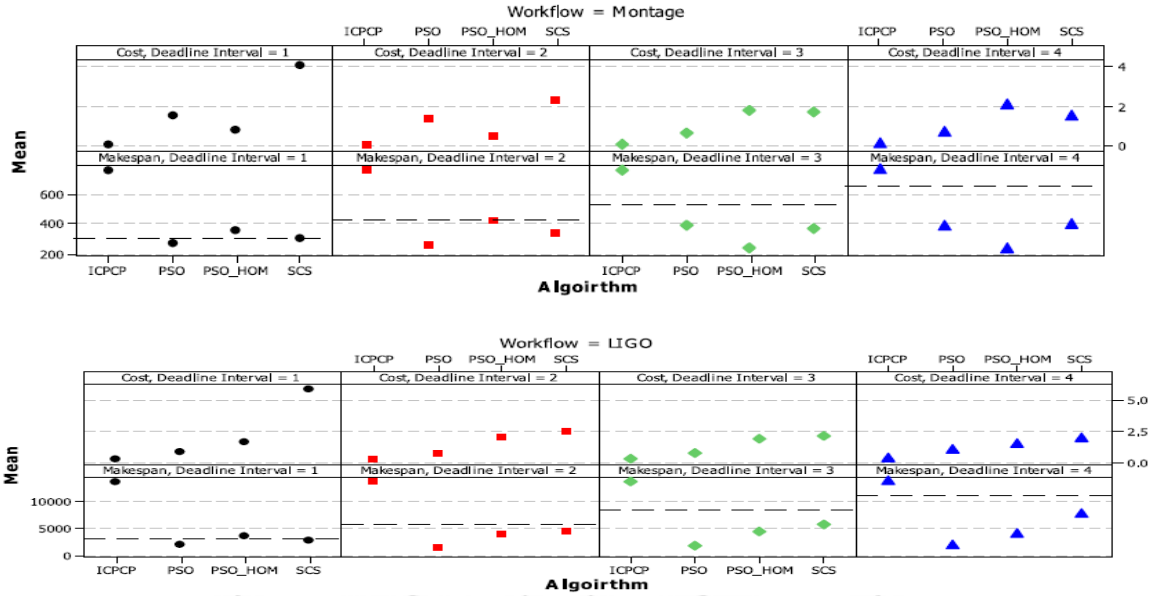

Figure 4: The reference line on each panel indicates the deadline value of the corresponding deadline interval.

For the Montage workflow, IC-PCP execution costs are the lowest ones for the four deadline intervals but its execution times are on average much higher than each of the four deadlines.

The performance of IC-PCP for the LIGO workflow is the same as for the SIPHT application; it achieves the lowest average cost in every case but produces the schedules with the longest execution times, which are well above the deadline value for the four intervals.

5. Conclusions and Future Work

In this paper proposes a system that includes minimize cost of the execution, resource provisioning, and deadline constrained scheduling scientific workflows on Infrastructure as a Service (IaaS) clouds. We use the concept of an algorithm based on the particle swarm optimization (PSO), deadline schedule generation. These algorithms minimize the overall workflow execution cost while meeting deadline constraints. This paper also focuses on IaaS clouds which provision a virtual machine pool of heterogeneous, unlimited resources that can be accessed based on user demand.

In future work, we would like to explore different options as it has a significant impact on the performance of the algorithm for the selection of the initial resource pool. We would also like to experiment with various optimization strategies such as genetic algorithms and compare their performance with art algorithm. Another future work is extending the resource model to consider that VMs can be deployed on different regions on the data transfer cost between data centers. Finally, we aim to implement our approach that it can be utilized for deploying applications in real life environments in a workflow engine.

References

[1] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K.Vahi, "Characterizing and profiling scientific workflows," Future Generation Comput. Syst., vol. 29.
[2] P. Mell, T. Grance, "The NIST definition of cloud computing recommendations of the NIST
[3] R. Buyya, J. Broberg, and A. M. Goscinski, Eds., Cloud Computing: Principles and Paradigms,
[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. 6th IEEE Int. Conf. Neural.
[5] Y. Fukuyama and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage stability," in Proc. 11th IEEE Int. Conf. Intell. Syst. Appl. Power Syst.
[6] C. O. Ourique, E. C. Biscaia Jr., and J. C. Pinto, "The use of particle swarm opt for dynamical analysis in chemical processes," Comput. Chem. Eng., vol. 26, no. 12, pp. 1783–1793, 2002.
[7] T. Sousa, A. Silva, and A. Neves, "Particle swarm based data mining algorithms for classification tasks," Parallel Comput., vol. 30, no. 5, pp. 767–783, 2004.
[8] M. R. Garey and D. S. Johnson, Computer and Intractability: AGuide to the NP-Completeness, vol. 238, New York, NY, USA:Freeman, 1979.
[9] M. Rahman, S. Venugopal, and R. Buyya, "A dynamic critical path algorithm for scheduling scientific workflow applications on global grids," in Proc. 3rd IEEE Int. Conf. e-Sci. Grid Comput.,
[10] W. N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling

problem with various QoS requirements," IEEE Trans. Syst., Man,Cybern Part C: Appl. Rev., vol. 39, no. 1, pp. 29–43, Jan. 09

[11] J Yu and R Buyya, "A budget constrained scheduling of workflow applications on utility grids using algorithms," in Proc. 1st Workshop Workflows Support Large-Scale Sci., 2006, pp. 1

[12] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in Proc. Int. Conf. High Perform. Comput., Netw., Storage., 2011, pp. 1–12.

[13] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," in Proc. Int. Conf. High Perform.

[14] A. Lazinica, Ed. Particle Swarm Optimization. Rijeka, Croatia: In Tech, 2009