© 2015 IJSRSET | Volume 1 | Issue 1 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Meeting of Time Limit Based Resource Distribution for Process in Cloud

R.Ramesh Kannan[1], S.Abinaya[2], D.Dheepikaraghavi[3]
Dhanalakshmi College of Engineering, Kancheepuram District, Tamilnadu, India

ABSTRACT

Cloud computing is the latest technology and it gives excellent possibilities to solve a systematic difficulties. It provides many queries that is used to finish the work economically. Even though it offers many benefits in workflow applications it also has some threats in cloud circumstances. In the existing invention the work get neglected due to the user's Quality of Service (QoS) and also it combines elasticity and heterogeneity as basic principles in computing assets. This paper presents resource provisioning and scheduling strategy for systematic workflows on Infrastructure as a Service (IaaS) Cloud. We use two algorithms namely meta-heuristic optimization technique and Particle Swarm Optimization (PSO), which plans to reduce the workflow execution cost in deadline constraints. Our heuristic is evaluated using systematic workflows and CloudSim. The conclusion of our project is, it advances better than the current state-of-the-art algorithms.

Keywords: Cloud Computing, Systematic Workflow, Resource Provisioning, Scheduling

I. INTRODUCTION

Workflows have been repeatedly used to model the systematic difficulties in areas such as physics, astronomy, and bioinformatics. Such systematic workflows have setting data and computing requirements and therefore demand a high performance computing surroundings in order to be executed in a required amount of time. The workflows are modeled by a set of tasks interconnected data or computing dependencies. They are studied by focusing on surroundings like Grids and Clusters. Although, with the emergence of new paradigms such as Cloud computing, novel approaches that address the particular challenges and opportunities of these technologies need to be developed.

Later the distributed environments have evolved from shared community platforms to utility-based models being the Cloud computing latest one. It enables the technology to deliver the IT resources over the Internet, and also follows a pay-as-you-go model where users are charged based on their consumption. There are many types of Cloud providers has different product offerings. They are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). This paper concentrates on IaaS Clouds which offer the user a virtual pool of unlimited, heterogeneous resources that can be accessed on demand. They also offer the flexibility of elastically acquiring or releasing resources with varying configurations which is suitable for requirements of an application. It empowers the users and gives them more control over the resources; it also explains the scheduling techniques so that the distributed resources are capably utilized.

While planning the execution of a Cloud environment there are two main stages. The first stage is the resources provisioning phase; during this stage, the computing resources that is used to run the tasks which are selected and provisioned. In the second one, the schedule is generated and each task is mapped onto the best suited resource. The selection of the resources and mapping of the tasks is done so that different user defined Quality of Service (QoS) requirements are met. The previous works in this area, are especially developed for Grids or Clusters, concentrated mostly on the scheduling phase. The reason behind this is that these surroundings provide a static pool of resources which are readily available to execute the tasks and whose configurations are known in advance. Both problems need to be addressed and

combined in order to produce an efficient execution plan, since this is not the case in Cloud environments.

Our work is based on the algorithms such as meta-heuristic optimization technique, Particle Swarm Optimization (PSO). PSO is inspired on the social behaviour of bird flocks. It is based on the swarm of particles moving through space and communicating with each other in order to determine an optimal search direction. PSO has high computational performance compared with other algorithms and parameters which makes easier to apply. In different areas many problems have been successfully addressed by using PSO to particular domains. This technique has been used to solve in areas such as reactive voltage control, pattern recognition and data mining among other areas.

We develop a static cost-minimization, deadline-constrained heuristic for scheduling a systematic workflow application in a Cloud environment. The fundamental features of IaaS providers are the dynamic provisioning and heterogeneity of computing resources and VM performance variation. To achieve this, both resources provisioning and scheduling are merged, modelled as an optimization problem. The contribution is, the algorithm with high accuracy in terms of meeting deadlines at low costs it considers heterogeneous resources that can be dynamically acquired and it is charged on a pay-peruse basis.

II. METHODS AND MATERIAL

RELATED CONCEPT

Workflows scheduling on scattered systems has been studied from the Multiprocessor Scheduling problem. So it is difficult to generate an optimal solution within polynomial time and algorithms focuses on generating approximate or real optimal solution. The Algorithms which are aim to find a schedule meets the users QoS requirements have been developed. The range of target surroundings in the proposed solution is similar or equal to community Grids. The limited pool of computing resources is assumed to be available and the execution cost is rarely a concern while minimizing the application's implementation time.
The solutions provide a valuable insight into the challenges and potential solutions for workflow scheduling. They are not optimal for utility like surroundings such as IaaS Clouds. There are many characteristics in Cloud atmosphere while developing the scheduling algorithm which is to be considered. For instance, Mao and Humphrey propose a dynamic approach for scheduling workflow ensembles on Clouds. There are various types of VMs with different prices and they can be leased on demand, depending on the application requirements. The execution cost is minimized based on the Clouds pricing model by which the VMs are paid by a fraction of time, which in most cases is one hour. The implementation cost is minimized by applying a set of heuristics such as merging tasks into a single one, identifying the most cost effective VM type for each task and instances. It is a suitable approach capable of reducing the completion cost of workflows on Clouds, the solution proposed only ensures a reduction on the cost and not a near-optimal solution.

The line with our work is presented by Abrishami which presents a static algorithm for scheduling a single workflow instance on an IaaS Cloud. Their algorithm is based on the workflow's partial critical paths and considers the Cloud features such as VM heterogeneity, pay-as-you-go and time interval pricing model. They also try to minimize the execution cost based on the heuristic of scheduling all tasks in a partial critical path on a single machine which finishes the tasks within the allocated time. They also have no global optimization technique and hence fail to utilize the whole workflow structure and characteristics to generate a better solution. Recently the algorithm estimates that the optimal number of resources that need to be leased to decrease the workflow execution cost. Their algorithm also generates a task to resource mapping and is designed to run online. The schedule and resources are updated every change time interval of the running VMs and tasks based on their current status. Their approach is the advantage of the elasticity of Cloud resources but it fails to consider the heterogeneous nature of the computing resources by assuming there is only one type of VM available.

III. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is an evolutionary computational method based on the behavior of animal herds. It was established by Eberhart and Kennedy in1995 and has been widely studied and utilized ever since. The algorithm is a stochastic optimization

technique in which the most basic concept is that of particle. A particle represents an separate (i.e. fish or bird) that has the skill to move through the distinct problem space and signifies a candidate solution to the optimization problem. At a specified point in time, the movement of particles is defined by their velocity, which is signified as a vector and therefore has magnitude and direction. This velocity is resolute by the finest position in which the particle has been so far and the best position in which any of the particles has been so far. Based on this, it is authoritative to be able to amount how decent (or bad) a particle's position is; this is attained by using a aptness function that measures the class of the particle's position and differs from problem to problem, depending on the framework and requirements.

Each particle is signified by its position and velocity. Particles keep path of their best position *pbest* and the global best position *gbest*; standards that are signified based on the fitness function. The algorithm will then at every step, change the velocity of each particle towards the *pbest* and *gbest* position. How much the particle transfers towards these values is biased by a random term, with different random numbers produced for acceleration towards *pbest* and gbest positions. The algorithm will continue to repeat until a stopping criterion is met; this is generally a specified maximum number of repetitions or a predefined fitness value considered to be good enough.

In each iteration, the position and velocity of a particle are rationalised based in Equations. The pseudo code for the algorithm is shown in Algorithm 1.

$$\overrightarrow{x_i}(t+1) = \overrightarrow{x_i}(t) + \overrightarrow{v_i}(t)$$
$$\overrightarrow{v_i}(t+1) = w.\overrightarrow{v_i}(t) + c_1 r_1(\overrightarrow{x_t^*}(t) - \overrightarrow{x_i}(t) + c_2 r_2(\overrightarrow{x^*}(t) - \overrightarrow{x_i}(t)$$

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.
4. Repeat from step 3 until the stopping criterion is met.

Where:

$$w = inertia$$
$$c_i = acceleration\ coefficient, i = 1,2$$
$$r_i = random\ number, i = 1,2\ and\ r_i \epsilon [0,1]$$
$$\overrightarrow{x_i^*} = best\ position\ of\ particle\ i$$
$$\overrightarrow{x^*} = position\ of\ the\ best\ particle$$
$$\overrightarrow{x_i} = current\ position\ of\ particle\ i$$

The velocity equation comprises various parameters that disturb the performance of the algorithm; moreover, some of them have a important influence on the convergence of the algorithm. One of these parameters is2v, which is recognized as the inertia factor or weight and is vital for the algorithm's convergence. This weight regulates how much previous velocities will influence the current velocity and states a trade-off between the indigenous cognitive component and global public experience of the particles. On one hand, a large inertia weight will make the velocity rise and therefore will favour overall exploration. On the other hand, a smaller value would make the particles slow down and hence favour local consideration. For this reason, a w value that stabilizes global and local search implies less iteration in order for the algorithm to converge.

Conversely, c1and c2do not have a serious effect in the convergence of PSO. However, spinning them properly may lead to a quicker convergence and may avert the algorithm to get trapped in local minima. Parameter c1is referred to as the cerebral parameter as the value c1r1in equation defines how much the preceding best position matters. On the other hand, c2is referred to as the common parameter as c1r2 in equations regulates the behavior of the particle relative to other neighbors.

There are other parameters that are not part of the velocity descriptions and are used as key to the algorithm. The first one is the number of particles; a larger value generally rises the likelihood of finding the

overall optimum. This number varies depending on the difficulties of the optimization problem but a typical span is between 20 and 40 particles. Other two parameters are the dimension of the particles and the range in which they are permitted to move, these values are solely determined by the kind of the problem being solved and how it is modeled to fit into PSO. Finally, the determined velocity defines the maximum change a particle can have in one repetition and can also be a parameter to the algorithm; however, this rate is usually set to be as large as the half of the location range of the particle.

IV. PROPOSED APPROACH

PSO Modeling

There are two key steps when modeling a PSO problem. The first one is describing how the problem will be fixed, that is, defining how the answer will be represented. The second one is defining how the "goodness" of a particle will be considered, that is, defining the aptness function.

To define the indoctrination of the problem, we need to found the sense and dimension of a particle. For the slating scenario presented here, a particle signifies a workflow and its tasks; thus, the dimension of the particle is equal to the number of chores in the workflow. The measurement of a particle will determine the coordinate system used to define its location in space. For example, the position of a 2-dimensional particle is stated by 2 coordinates, the position of a 3-dimensional one is stated by 3 coordinates and so on. As an example, the particle depicted in Figure represents a workflow with 9 tasks; the particle is a 9-dimensional one and its location is defined by 9 coordinates, coordinates 1 through 9.

The variety in which the particle is recognized to move is determined in this case by the number of resources available to run the chores. As a result, the worth of a coordinate can range from 0 to the number of VMs in the initial resource pool. Based on this, the numeral part of the value of each coordinate in a particle's location corresponds to a resource directory and represents the compute resource assigned to the task defined by that specific coordinate.

In this way, the particle's position encodes a mapping of task to resources .Following the example given in Figure; there are 3 resources in the resource tarn so each coordinate will have a value among 0 and 3. Coordinate 1corresponds to task 1 and its worth of 1.2 means that this task was allocated to resource 1. Coordinate 2 parallels to task 2 and its value of 1.0 specifies that task 2 was allocated to resource 1. The same reason applies to the rest of the coordinates and their values.
Since the fitness function is used to establish how good a possible solution is, it needs to replicate the objectives of the planning problem. Based on this, the aptness function will be reduced and its value will be the total implementation cost TEC; associated to the schedule S resulting from the particle's position.

Because of the elasticity and dynamicity of the resource attainment model offered by IaaS providers, there is no first set of available resources we can custom as an input to the algorithm. Instead, we have the delusion of an unlimited pool of varied VMs that can be acquired and released at any outlet in time.

Consequently, a tactic to define a first pool of resources that the algorithm can use to travel different solutions and accomplish the scheduling objective needs to be put in place.
Such strategy needs to return the heterogeneity of the VMs and give PSO enough choices so that a suitable particle (i.e. solution) is generated. If this first resource pool is limited, then so will be the resources that can be used to schedule the jobs. If it is very large, then the number of feasible schedules becomes very huge and so does the search space surveyed by PSO, making it hard for the algorithm to converge and discover a suitable solution.

As for the problem restraints, PSO was not designed to solve forced optimization problems. To address this, we use a form of PSO that incorporates the constraint-handling strategy projected by Deb et al. In such strategy, whenever two solutions are being equated ,the following rules are used to select the superior one. If both of the solutions are viable, then the solution with healthier fitness is selected. If on the other hand, one solution is feasible and the other one is not, then the feasible one is selected. Finally, if both solutions are infeasible, the one with the smaller overall constraint defilement is selected. The latter scenario suggests that a measure of how much

a solution violates a constraint requests to be in place. Our problem specifies a particular constraint, meeting the application's deadline. Therefore, we define the complete constraint violation value of a solution to be the variance between the solutions makes span and the workflow's deadline. In this way, a solution whose make span is nearer to the deadline will be preferred over a solution whose make span is further away.

V. CONCLUSIONS

In this paper we presented a collective resource provisioning and scheduling strategy for executing scientific workflows on IaaS Clouds. The scenario was showed as an optimization problem which aims to reduce the overall execution cost while meeting a consumer defined deadline and was resolved using the meta-heuristic optimization algorithm, PSO. The proposed approach includes basic IaaS Cloud principles such as a pay-as-you-go model, heterogeneity, elasticity, and dynamicity of the resources.

Furthermore, our solution reflects other characteristics typical of IaaS platforms such as performance variation and VM boot time.

VI. REFERENCES

[1]. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., & Vahi, K. (2012). Characterizing and profiling scientific workflows. Future Generation Comput. Syst. 29(3), 682- 692.

[2]. Mell, P., and T. Grance. (2011). The NIST definition of cloud computing—recommendations of the National Institute of Standards and Technology. Special Publication 800-145, NIST, Gaithersburg.

[3]. Buyya, R., Broberg, J., and Goscinski, A. M. (Eds.). (2010). Cloud computing: Principles and paradigms (Vol. 87).

[4]. Wiley.Kennedy, J., and Eberhart, R. (1995). Particle swarm optimization. In Proc. 6th IEEE Int. Conf. Neural Networks, 1942-1948.

[5]. Fukuyama, Y., and Nakanishi, Y. (1999). A particle swarm optimization for reactive power and voltage control consider- ing voltage stability. In Proc. 11th IEEE Int. Conf. Intelligent Systems Application to Power Systems (ISAP), 117-121.

[6]. Ourique, C. O., Biscaia Jr, E. C., and Pinto, J. C. (2002). The use of particle swarm optimization for dynamical analysis in chem- ical processes. Comput. & Chemical Eng., 26(12), 1783-1793.

[7]. Sousa, T., Silva, A., and Neves, A. (2004). Particle swarm based data mining algorithms for classification tasks. Parallel Computing, 30(5), 767-783.

[8]. Garey, M. R., & Johnson, D. S. (1979). Computer and intractability: A Guide to the NP-Completeness. Ney York, NY. WH Freeman and Company. 238.

[9]. Rahman, M., Venugopal, S., and Buyya, R. (2007). A dynamic critical path algorithm for scheduling scientific workflow applications on global grids. In Proc. 3rd IEEE Int. Conf. e-Sci. and Grid Computing, 35-42.

[10]. Chen, W. N., and Zhang, J. (2009). An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. IEEE Trans. Syst., Man, Cybern., Part C: Applicat. Reviews, 39(1), 29-43.

[11]. Yu, J., and Buyya, R. (2006). A budget constrained scheduling of workflow applications on utility grids using genetic algorithms. In Proc. 1st Workshop on Workflows in Support of Large-Scale Sci. (WORKS), 1-10.

[12]. Mao, M., and Humphrey, M. (2011). Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In Proc. Int. Conf. High Performance Computing, Networking, Storage and Analysis (SC), 1-12.

[13]. Malawski, M., Juve, G., Deelman, E., and Nabrzyski, J. (2012). Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. In Proc. Int. Conf. High Performance Computing, Networking, Storage and Anal. (SC), 22.

[14]. Abrishami, S., Naghibzadeh, M., and Epema, D. (2012). Dead- line-constrained workflow scheduling algorithms for IaaS Clouds. Future Generation Comput. Syst., 23(8), 1400-1414.

[15]. Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In Proc. IEEE Int. Conf. Advanced Inform. Networking and Applicat. (AINA), 400- 407.

[16]. Wu, Z., Ni, Z., Gu, L., & Liu, X. (2010). A revised discrete parti-cle swarm optimization for cloud workflow scheduling. In Proc. IEEE Int. Conf. Computational Intell. and Security (CIS), 184-188.

[17]. Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., and Epema, D. (2010). A performance analysis of EC2 cloud computing services for scientific computing. In Cloud Compu- ting. 115-131. Springer Berlin Heidelberg.